On the difficulty of computing the winners of a tournament

Olivier Hudry

On the difficulty of computing the winners of a tournament

Olivier Hudry*

Abstract

In voting theory, the result of a paired comparison method as the one suggested by Condorcet can be represented by a tournament, i.e., a complete asymmetric directed graph. When there is no Condorcet winner, i.e., a candidate preferred to any other candidate by a majority of voters, it is not always easy to decide who is the winner of the election. Different methods, called tournament solutions, have been proposed to define the winners. They differ by their properties and usually lead to different winners. Among these properties, we consider in this survey the algorithmic complexity of the most usual tournament solutions: for each one of these methods, we give its complexity.

Keywords : voting theory, tournament solutions, majority tournament, Copeland solution, maximum likelihood, self-consistent choice rule, Markovian solution, uncovered set, Banks solution, Slater solution, complexity, polynomial problems, NP-hardness

1 Introduction

A tournament $T = (X, A)$ is a complete asymmetric directed graph: for any vertex x and any vertex y with $x \neq y$, there exists exactly one of the two arcs (i.e., oriented edges) (x, y) or (y, x). This kind of graphs may represent for instance the result of a paired comparison method, as the one suggested by Condorcet [4] at the end of the 18^{th} century as a voting procedure. In such a situation, X is the set of candidates; for each pair $\{x, y\}$ of distinct candidates, we compute the number m_{xy} of voters who prefer x to y; if we

*École nationale supérieure des télécommunications and CNRS, 46, rue Barrault, 75634 Paris cedex 13, France. hudry@enst.fr

have $m_{xy} > m_{yx}$, x is preferred to y by a majority of voters. It is usual to represent the result of this method by a graph $T = (X, A)$ of which the vertex set is the set of candidates and such that, for any vertex x and any vertex y with $x \neq y$, there is an arc (x, y) if x is preferred to y by a majority of voters (i.e. if $m_{xy} > m_{yx}$). If there is no tie (or, equivalently, if we have a tie-breaking rule), then $T = (X, A)$ is a tournament, called the *majority tournament*. In the following, we keep this illustration arising from the voting theory, though paired comparison methods may occur in other contexts (sports, psychology, statistics, and so on).

If the tournament T is transitive, then it is a linear order which gives a collective ranking of the candidates. Even if T is not transitive, it may happen that a vertex x is collectively preferred to any other vertex; from a graph theoretic point of view, it means that all the arcs adjacent to x go from x towards the other vertices or, equivalently, that the out-degree of x is equal to $n-1$, where n denotes the number of vertices of T. When such a vertex exists, there are good reasons to consider it as the winner. Such a winner is called a *Condorcet winner*. If a Condorcet winner exists, it is unique. When there is no Condorcet winner in T, deciding who is the winner of the election may be a difficult task.

Different methods, called *tournament solutions* have been proposed to compute such a winner (see [16] for the definitions and the properties of the tournament solutions considered below). They differ by their axiomatic properties and usually lead to different winners. In this paper, we study the complexities of the most usual tournament solutions. From a practical point of view, the complexity of a method plays an essential role; for instance for an election, it is quite important to be able to declare who is the winner in a "reasonable" time. With this respect, polynomial methods (i.e., methods of which the algorithmic complexity is upper bounded by a polynomial function in the size of the data) appear usually as preferable to exponential methods (i.e., methods of which the algorithmic complexity is not upper bounded by a polynomial function in the size of the data), though this complexity is expressed for the worst case. When a problem is NP-hard (or NP-complete if we deal with a decision problem, i.e. a problem in which a question is set with "yes" or "no" as its answer), the only methods known nowdays to solve the problem exactly are exponential. Hence the interest in the complexity of the tournament solutions (for the theory of NP-completeness, see for instance [10] or [3]).

2 Tournament solutions and their complexities

We give now the definitions of the considered tournament solutions and their complexities. In the sequel, $T = (X, A)$ will denote a tournament; n will denote the number of vertices of T; the out-degree of any vertex $x \in X$ is called the *(Copeland) score* of x and is noted $s(x)$. When T is not connected, it is possible to decompose it into its strongly connected components; because of the structure of tournament, there exists a strongly

connected component, sometimes called the *Top Cycle* $TC(T)$ of T, such that all the arcs with one extremity inside $TC(T)$ and the other outside are oriented from $TC(T)$ towards $X \setminus TC(T)$. It is easy to compute $TC(T)$ polynomialy (more precisely in $O(n^2)$) for instance thanks to the application of a depth-first search procedure (see for instance [7]). Notice that, when a Condorcet winner exists in T, $TC(T)$ contains only this vertex; in this case, all the solutions below select the Condorcet winner as their unique winner. More generally, all the tournament solutions considered below select the winners of T in $TC(T)$; so we may restrict ourselves in searching the winners inside $TC(T)$, which induces a strongly connected subtournament of T. For this reason, we assume from now on that T is strongly connected (as a consequence, T does not admit any Condorcet winner).

Notice that, as the number of arcs of T is equal to $n(n-1)/2$, we need at least $n(n-1)/2$ bits to encode T. On the other hand, encoding T by its adajency matrix requires n^2 bits; thus, the (binary) size of T is $\Theta(n^2)$ and we may consider in the following that T is encoded by its adjacency matrix. So a method is polynomial with respect to the size of the instance given by T if and only if it is polynomial with respect to n. It is with respect to this parameter n that we are going to state the complexity of the tournament solutions described below.

2.1 Maximum scores: Copeland solution

An easy method to find the winners is the solution proposed by Copeland ([6]), leading to the *Copeland winners*.

Definition 1 *A Copeland winner of the tournament $T = (X, A)$ is any vertex with a maximum score.*

Then Copeland solution is obviously polynomial:

Theorem 1 *Copeland winners can be computed in $O(n^2)$.*

Proof. To get the Copeland winners of T, it is sufficient to compute the scores of the n vertices of T, what can be done in $O(n^2)$ by summing up the binary values of each row of the adjacency matrix of T, then to compute the maximum value of these scores, what can be done in $O(n)$, last to compare each score to this maximum value, what can be done also in $O(n)$. Hence the result. ◇

Notice that this method can give all the Copeland winners with the same complexity. Notice also that, if we assume that the out-degrees, i.e. the scores, are known, then the complexity is only $O(n)$. Last, notice also that ranking the vertices according to the decreasing values of the scores gives a preorder P of which the maximal elements are the

Copeland winners; if we want to rank the vertices according to a linear order, then we may adopt any linear extension of P; in this case anyway, the number of rankings may be very large (for instance,when n is odd, if all the scores are equal to $(n-1)/2$, then the $n!$ linear orders are relevant rankings) and the method is no longer polynomial.

2.2 Maximum likelihood: Zermelo solution

Zermelo designed the following method in 1929 [22]. A positive "strength" $\sigma(x)$ is associated with each vertex x; for instance, $\sigma(x)$ may be interpreted as the popularity rating of the candidate x. Assume that these strengths are known. Then it would be natural to rank the candidates according to the decreasing values of these strengths, and the candidates with the greatest strength would be the Zermelo winners. So, the question is: how to compute these strengths ?

To answer this question, Zermelo considers that the probability $p(x,y)$ that a candidate x is preferred to a candidate y by a majority of voters is given by

$$p(x,y) = \frac{\sigma(x)}{\sigma(x)+\sigma(y)}.$$

If we assume moreover that these collective preferences are independent the ones from the others when we consider all the possible pairs of candidates (what is not obviously the case in practice), then the conditional probability $p(T/\sigma)$ to get $T=(X,A)$ as the majoritary tournament knowing σ is given by:

$$p(T/\sigma) = \prod_{(x,y)\in A} \frac{\sigma(x)}{\sigma(x)+\sigma(y)}.$$

So, the maximum likelihood method proposed by Zermelo consists in computing the positive strengths $\sigma(x)$ for $x \in X$ maximizing $p(T/\sigma)$ with $\sum_{x\in X}\sigma(x)=1$.

Though the strengths are not necessarily proportional to the Copeland scores, it is possible to show that the Zermelo winners and the Copeland winners are the same (more precisely, the ranking induced by the strenghts is the same as the ranking induced by the scores). Consequently, because of the previous theorem, we get:

Theorem 2 *Zermelo winners can be computed in $O(n^2)$ but the enumeration of all the rankings compatible with the maximum likelihood rule may require an exponential time.*

2.3 Self-consistent choice rule, or Markovian solution

The self-consistent choice rule is due to Levchenkov [17], [11], but the following presentation has been proposed by Laslier [15]. We start with a vertex x_0 randomly chosen, and

we generate a series $\{x_k\}$ of elements belonging to X as follows. At each step k, a vertex x is randomly chosen with a uniform distribution over X. If (x, x_k) is an arc of T, then we set $x_{k+1} = x$; otherwise, we keep x_k: $x_{k+1} = x_k$. Then we repeat the same process. This random walk leads to the different vertices with different probabilities, depending on their "attractiveness". These probabilities may be used to sort the vertices and thus to define the winners.

More precisely, we way associate a Markov chain with this random walk. The graph $G = (X, B)$ describing this Markov chain has the same vertex set as T, i.e. X, but the arcs of G, defining the possible transitions from the current state (i.e., the current vertex) to another one, are exactly the arcs which do not belong to T, including the loops: $B = X \times X \setminus A$. The transition probabilities p are defined as follows: for $x \in X$ and $y \in X$ with $x \neq y$ and $(x, y) \in B$ (hence $(y, x) \in A$), the transition probability $p(x, y)$ to go from x to y is equal to $\frac{1}{n-1}$; for a loop (x, x), the transition probability $p(x, x)$ to stay on x is equal to $\frac{s(x)}{n-1}$, where $s(x)$ still denotes the score of x in T. Then, at each step k of this process, we may define a probability distribution vector π_k giving the probability to be on each vertex of T: for any integer $k \geq 0$ and any vertex $x \in X$, $\pi_k(x)$ gives the probability to be on vertex x at step k. Let $P = (p_{xy})_{(x,y)\in X^2}$ denote the matrix of the transition probabilities: $p_{xy} = p(x, y)$ if $(x, y) \in B$ and $p_{xy} = 0$ otherwise. Then the expression of π_{k+1} for $k \geq 0$ is $\pi_{k+1} = \pi_k \times P$, or also $\pi_{k+1} = \pi_0 \times P^{k+1}$, where π_0 is the vector with only 0's as components, except for the component associated with the starting vertex x_0, equal to 1.

The theory of Markov chains [8] shows that, as T is assumed to be strongly connected, the series of the probability distributions π_k's tends towards a limit π^*, and this limit is independent of π_0. This distribution allows to define the self-consistent choice rule:

Definition 2 *The self-consistent choice winners of T (i.e., the winners according to the self-consistent choice rule) are the vertices x^* verifying:*

$$\pi^*(x^*) = \max_{x \in X} \pi^*(x).$$

Lemma 1 *The self-consistent choice winners can be computed within the same complexity as the multiplication of two $(n \times n)$-matrices.*

Proof. Still from the theory of Markov chains [8], it is easy to show that π^* satisfies the equality $\pi^* = \pi^* \times P$. This equality is obviously not sufficient to characterize the values of $\pi^*(x)$ for all the vertices x. But, because T is strongly connected, the linear system defined by $\pi^* = \pi^* \times P$ and $\sum_{x \in X} \pi^*(x) = 1$ admits a unique solution. Moreover, still because T is strongly connected, we may remove any row of the system $\pi^* = \pi^* \times P$ in such a way that the new system given by the remaining $n - 1$ rows, along with the

equation $\sum_{x \in X} \pi^*(x) = 1$ still admits a unique solution. It yields that the computation of the self-consistent choice winners may be done by the resolution of a linear system with n variables and n equations admitting a unique solution. Hence the complexity of the self-consistent choice rule, since the resolution of such a linear system has the same complexity as the multiplication of two $(n \times n)$-matrices (see [7] for instance). $\diamond$

Theorem 3 *The self-consistent choice winners can be computed in* $O(n^{2.38})$.

Proof. The complexity $O(n^{2.38})$ is a consequence of the previous lemma and of the fact that it is possible to multiply two $(n \times n)$-matrices in $O(n^{2.38})$ [7]. $\diamond$

2.4 The uncovered set

Let x and y be two distinct vertices. We say that x *covers* y if any successor of y is also a successor of x:

$$\forall z \in X, (y, z) \in A \Rightarrow (x, z) \in A.$$

Notice that, because of the asymmetry of a tournament (what involves that T is irreflexive), if x covers y, then (x, y) is an arc of T. A vertex is said to be *uncovered* if none vertex covers it; the uncovered set of T is noted $UC(T)$. Adopting the elements of $UC(T)$ as the winners of T has been independently suggested by Fishburn in 1977 [9] and by Miller in 1980 [18]. This method is polynomial:

Lemma 2 *Computing the uncovered elements can be done within the same complexity as the multiplication of two* $(n \times n)$*-matrices.*

Proof. It is well-known that a vertex x is uncovered if and only if, for any other vertex y, there exists a directed path with one or two arcs from x to y (it is the so-called "two steps principle", see [19]). If M denotes the adjacency matrix of T, the entries $m^2_{x,y}$ of M^2 give the numbers of paths with two arcs from x to y. Thus, to know whether x is uncovered, it is sufficient to compute the row of $M^2 + M + I$ (where I denotes the identity matrix) associated with x: x is uncovered if and only of there is no entry of this row equal to 0. As the complexity of this process is the same as the one of the computation of M^2 (the other steps are negligible), we may compute simultaneously all the uncovered vertices within the same complexity as the multiplication of M by itself. $\diamond$

Theorem 4 *Computing the uncovered elements can be done in* $O(n^{2.38})$.

Proof. The complexity $O(n^{2.38})$ is a consequence of the previous lemma and of the fact that it is possible to multiply two $(n \times n)$-matrices in $O(n^{2.38})$ [7]. $\diamond$

A variant of this solution consists in computing the series $UC^k(T)$ defined as follows: $UC^1(T)$ is equal to $UC(T)$; for $k > 0$, $UC^{k+1}(T)$ is the uncovered set of the subtournament of T induced by $UC^k(T)$. The elements belonging to $UC^k(T)$ for $2 \leq k \leq n$ (notice that the sets $UC^k(T)$ cannot evolve any longer for exponents greater than n) may be proposed as the winners of T (though some basic properties are not satisfied by this method; see [15] or [16]). Because of the polynomiality of UC and as it is possible to build a subtournament induced by a subset of vertices in polynomial time, we get the following result about the generalization of UC:

Theorem 5 *For $1 \leq k \leq n$, computing the winners according to UC^k can be done in polynomial time.*

2.5 Maximal transitive subtournaments: Banks solution

As said above, when T is transitive, there exists a unique Condorcet winner, and it is quite natural to select it as the winner of T (notice anyway that there exist many voting procedures which do not necessarily select the Condorcet winner, when such a winner exists; it is the case for instance for the procedure applied in France to elect the President of the Republic, or usually applied for the election of the members of the French parliament). When T is not transitive, we may anyway consider the transitive subtournaments of T which are maximal with respect to inclusion, and then select their Condorcet winners as the winners of T. This defines the Banks solution [2]:

Definition 3 *A Banks winner of T is the Condorcet winner of any maximal (with respect to inclusion) transitive subtournament of T.*

From the complexity point of view, this solution owns a maybe unexpected property. Indeed, G. Wöginger recently shows the following theorem [21]:

Theorem 6 *The following problem is NP-complete:*

Instance: a tournament T, a vertex x of T;

Question: is x a Banks winner of T?

Anyway the next theorem is proved in [12]:

Theorem 7 *For any tournament T, computing a Banks winner is polynomial, and more precisely, can be done in $O(n^2)$.*

There is no paradox: when we compute a Banks winner, we do not (and cannot, in the general case, if P and NP are not equal) choose the Banks winner that we would like to get. Moreover, as there are at most n Banks winners and because of G. Woeginger's result, we get:

Theorem 8 *Computing all the Banks winners of T is an NP-hard problem.*

Proof. It is easy to design Turing transformations showing that, on one hand, checking that a given vertex is a Banks winner, and, on the other hand, enumerating all the Banks winners are polynomialy linked. Indeed, if we can enumerate all the Banks winners with some complexity, then we may obviously check whether a given vertex is a Banks winner within the same complexity. Conversely, if we can check whether a given vertex is a Banks winner thanks to an algorithm with some complexity $c(n)$, then we may enumerate all the Banks winners by applying this algorithm to the n vertices of the tournament: this gives an algorithm to solve the problem of the enumeration with $n.c(n)$ as its complexity. Because of this link (in fact, because of the first part of this link) and because of the NP-completeness of the problem consisting in checking whether a given vertex is a Banks winner, the problem of the enumeration of all the Banks winners is NP-hard. ◇

2.6 Linear orders at minimum distance: Slater solution

The last solution described here consists in reversing a minimum number of arcs of T in order to get a transitive tournament O, i.e. a linear order, and then to consider the Condorcet winner of O. This defines a *Slater winner* [20]:

Definition 4 *Let O be a linear order defined on X. We define the distance $d(T, O)$ between T and O as the number of arcs of T which have a different orientation in O. A Slater order of T is a linear order O^* minimizing $d(T, O)$ over the set of the linear orders O defined on X. A Slater winner of T is the Condorcet winner of any Slater order of T. We note $i(T)$ the minimum number of arcs that must be reversed in T to get a Slater order O^* of T: $d(T, O^*) = i(T)$.*

The complexity of Slater solution may be derived from a recent result dealing with a problem called *the Feedback Arc Set Problem*. This problem consists, given a directed graph G, in removing a minimum number of arcs from G in order to get a graph without any circuit. From the work by Karp [14], this problem is known to be NP-complete in the general case. Recent results ([1] and [5]) show that this problem remains NP-complete even when restricted to tournaments. From this, we may prove the following theorem (see [13] for details):

Theorem 9 *For any tournament T, we have the following results:*

1. the computation of $i(T)$ is NP-hard;

2. the computation of a Slater winner of T is NP-hard;

3. checking that a given vertex is a Slater winner of T is NP-hard;

4. the computation of a Slater order of T is NP-hard;

5. the computation of all the Slater winners of T or all the Slater orders of T is NP-hard.

3 Conclusion

We may summarize the previous results as follows:

– some tournament solutions are polynomial with respect to the size of T: it is the case for the Copeland solution (maximum scores), the Zermelo solution (maximum likelihood), the self-consistent choice rule or Markovian solution (Levchenkov, Laslier), and the uncovered set (Fishburn, Miller);

– Banks solution (maximal transitive subtournaments) is polynomial if we consider the computation of one (not specified) Banks winner, but NP-hard if we consider the problem consisting in checking that a given vertex is a Banks winner or the problem consisting in enumerating all the Banks winners;

– Slater solution (linear order at minimum distance) is NP-hard for its different variants.

From a practical point of view in the context of voting theory, these results give an advantage to the first methods, if we want to get the issue of the election in a “reasonable” time.

References

[1] Alon N. (2006), Ranking tournaments, SIAM J. Discrete Math. 20, 137-142.

[2] Banks J. (1985), Sophisticated voting outcomes and agenda control, Social Choice and Welfare, 2, 295-306.

[3] Barthélemy J.-P., Cohen G., Lobstein A. (1996), Algorithmic Complexity and Communication Problems, UCL Press, London.

[4] Caritat M.J.A.N., marquis de Condorcet (1785), Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, Paris.

[5] Charbit P., Thomasse S., Yeo, A., The minimum feedback arc set problem is NP-hard for tournaments, to appear in Combinatorics, Probability and Computing.

[6] Copeland A.H. (1951), A "reasonable" social welfare function, Seminar on applications of mathematics to social sciences, University of Michigan.

[7] Cormen T., Leiserson C., Rivest R. (1990), Introduction to algorithms, MIT Press, Cambridge, Massachusets.

[8] Durrett R. (1991), Probability: theory and examples, Duxbury, Belmont.

[9] Fishburn, P. (1977), Condorcet social choice functions, SIAM Journal of Applied Mathematics, 33, 469-489.

[10] Garey M.R., Johnson D.S. (1979), Computers and intractability, a guide to the theory of NP-completeness, Freeman, New York.

[11] Grivko I. L., Levchenkov V.S. (1994), Intrinsic properties of the self-consistent choice rule, Automation and Remote Control 55, 689-697.

[12] Hudry O. (2004), A note on "Banks winners in tournaments are difficult to recognize" by G.J. Woeginger, Social Choice and Welfare 23, 1-2.

[13] Hudry O., Complexities of Slater problems, in preparation.

[14] Karp R.M. (1972), Reducibility among combinatorial problems, in R.E. Miller and J.W. Tatcher (eds.), Complexity of computer computations, New York, Plenum Press, 85-103.

[15] Laslier J.-F. (1993), Solutions de Tournois, Habilitation à diriger les recherches en science économique, université de Cergy-Pontoise.

[16] Laslier J.-F. (1997), Tournament Solutions and Majority Voting, Springer, Berlin, Heidelberg, New York.

[17] Levchenkov V.S. (1992), Social choice theory: a new sight, Preprint of the Institute for System Analysis, Moscou.

[18] Miller, N. (1980), A new solution set for tournaments and majority voting: Further graph-theoretical approaches to the theory of voting, American Journal of Political Science, 24 (1), 68-96.

[19] Shepsle K., Weingast B. (1984), Uncovered sets ans sophisticated voting outcomes with implications for agenda institutions, American Journal of Political Sciences 28, 49-74.

[20] Slater P. (1961), Inconsistencies in a schedule of paired comparisons, Biometrika, 48, 303-312.

[21] Woeginger G. J. (2003), Banks winners in tournaments are difficult to recognize, Social Choice and Welfare 20(3), 523-528.

[22] Zermelo E. (1929), Die Berechnung der Turnier-Ergebnisse als ein maximal Problem der Warscheinlichkeistsrechnung, Math. Zeitung, 29, 436-460.