European Journal of Operational Research 203 (2010) 216–221

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

On the complexity of Slater's problems

Olivier Hudry

École nationale supérieure des télécommunications 46, rue Barrault, 75634 Paris Cedex 13, France

ARTICLE INFO

Article history:
Received 23 October 2008
Accepted 28 July 2009
Available online 5 August 2009

Keywords:
Complexity
Tournament solutions
Slater solution
Majority tournament

ABSTRACT

Given a tournament T, Slater's problem consists in determining a linear order (i.e. a complete directed graph without directed cycles) at minimum distance from T, the distance between T and a linear order O being the number of directed edges with different orientations in T and in O. This paper studies the complexity of this problem and of several variants of it: computing a Slater order, computing a Slater winner, checking that a given vertex is a Slater winner and so on.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

A *tournament T* is an asymmetric and complete directed simple graph: between two distinct vertices x and y, there is one and only one of the two *arcs* (i.e. directed edges) (x,y) and (y,x). Such a structure arises for instance in voting theory, to model the result of a pairwise comparison method, as the one suggested by Condorcet [11]. Indeed, assume that we are given a set X of n candidates and a collection $\Pi = (O_1, O_2, \ldots, O_m)$, called a *profile*, of the opinions O_i of m voters $(1 \leqslant i \leqslant m)$ who want to rank the n candidates; we assume moreover that the individual preferences O_i $(1 \leqslant i \leqslant m)$ of the m voters are linear orders over X. An order O defined on X will be represented under the form $x_1 > x_2 > \cdots > x_n$ for an appropriate numbering of the n elements x_j $(1 \leqslant j \leqslant n)$ of X, with the agreement that $x > y$ means that x is preferred to y; with this notation, we say that x_1 is *the first element of O*. More generally, if O and O' denote two linear orders defined on different sets X and X', $O > O'$ will denote the linear order defined on $X \cup X'$ as the concatenation of O followed by O'.

In order to aggregate these m linear orders into a linear order which can be considered as the collective preference, Condorcet suggested to compute, for each pair of candidates $\{x,y\}$ (with $x \neq y$), the number m_{xy} of voters who prefer x to y and the number m_{yx} of voters who prefer y to x; x is then collectively preferred to y if we have $m_{xy} > m_{yx}$. We may model this collective preference by a graph T, with the set of candidates as its set of vertices, and with an arc from x to y when x is collectively preferred to y, i.e. when we have $m_{xy} > m_{yx}$. If there is no tie (it is in particular the case when m is odd, since we have $m_{xy} + m_{yx} = m$), as it will be assumed in the sequel, the obtained graph $T = (X, A)$ is a tournament, called the *majority tournament* of the election, where X is the set of candidates and where A is thus defined by: $(x,y) \in A \iff m_{xy} > m_{yx}$ (see [9] and the references therein for more details).

Sometimes T is a linear order O, i.e. a complete directed graph without directed cycles, providing the desired ranking of the candidates of the election. It is well-known (see [6,20,21,22,23] for the definitions and the basic results about graphs and tournaments) that a linear order is a transitive tournament and conversely (remember that a tournament T is transitive if the existence in T of the arcs (x,y) and (y,z) implies the one of (x,z)), and that a tournament is transitive if and only if it is without any *circuit* (i.e. any directed cycle). But, as discovered by Condorcet himself, the majority tournament $T = (X, A)$ of an election may not be transitive, even if the preferences of the voters are all assumed to be linear orders. This is the so-called "voting paradox" or also "Condorcet effect" [14]. The following example illustrates this situation.

Example 1. Assume that m=9 voters must rank $n = 4$ candidates a, b, c, and d. The profile Π_0 of the preferences of the voters is assumed to be given by the following linear orders:

- the preferences of three voters are: $a > b > c > d$;
- the preferences of two voters are: $b > d > c > a$;
- the preference of one voter is: $c > d > a > b$;
- the preference of one voter is: $d > a > c > b$;
- the preference of one voter is: $d > b > a > c$;
- the preference of one voter is: $c > d > b > a$.

Thus we have $\Pi_0 = (a > b > c > d,\ a > b > c > d,\ a > b > c > d,\ b > d > c > a,\ b > d > c > a,\ c > d > a > b,\ d > a > c > b,\ d > b > a > c,\ c > d > b > a)$. The quantities m_{xy} implied in

E-mail address: hudry@enst.fr

Condorcet's procedure are the following, where the bold values show the ones greater than or equal to the strict majority $(m+1)/2$, here equal to 5:

- $m_{ab} = \mathbf{5}; m_{ba} = 4;$
- $m_{ac} = \mathbf{5}; m_{ca} = 4;$
- $m_{ad} = 3; m_{da} = \mathbf{6};$
- $m_{bc} = \mathbf{6}; m_{cb} = 3;$
- $m_{bd} = \mathbf{5}; m_{db} = 4;$
- $m_{cd} = \mathbf{5}; m_{dc} = 4.$

Here, the majority tournament is not a linear order, but the tournament T_0 of Fig. 1.

Even if T is not transitive, it may happen that there exists a *Condorcet winner*: a Condorcet winner is a candidate ζ collectively preferred to any other candidate: $\forall\, x \in X$ with $x \neq \zeta,\ m_{\zeta x} > m_{x\zeta}$. From the graph theoretic point of view, it means that all the arcs involving ζ go from ζ to the other vertices: the out-degree of ζ is equal to $n-1$ and its in-degree is equal to 0. Notice that, when there exists a Condorcet winner, there is only one.

When there exists a Condorcet winner of the election, we may consider him as the winner of the election (though it is not the case for many voting procedures). When there is no Condorcet winner (as it is the case for the example above), the question arises to know who can be considered as the winner of the election. Several methods, known under the name of *tournament solutions*, have been designed to answer this question (see for instance [20] for a survey on these solutions and [17] for their complexities). Among them, we find for example the solution proposed by Banks [3]. It consists in considering as winners (called the *Banks winners* of T) the first elements of the maximal (with respect to inclusion) transitive subtournaments of T (for instance, the Banks winners of the tournament of Fig. 1 are a because of the maximal transitive subtournament $a > b > c$, b because of the maximal transitive subtournament $b > c > d$, and d because of the maximal transitive subtournament $d > a$).

Another solution, that we are going to consider in this paper, is the one studied by P. Slater, called *Slater's solution* ([24]; for a survey and references on this problem, see [9,10]). Slater's solution consists in transforming Tinto a linear order by reversing a minimum number of arcs of T. For instance, it is easy to see that it is necessary and sufficient to reverse the arc (d, a) in the tournament of Fig. 1 to transform it into the linear order $a > b > c > d$.

More formally, in the sequel, $T = (X, A)$ will denote a tournament with n vertices. The set of linear orders defined on X will be $\omega(X)$, and O will represent a linear order defined on X (in other words, O is an element of $\omega(X)$). We define the *symmetric difference distance* δ (in fact, half this distance) between T and $O = (X, B)$ by:

$$\delta(T, O) = \frac{1}{2}|A \Delta B|,$$

where Δ denotes the usual *symmetric difference* between sets. As T and O are complete and asymmetric, we have also:

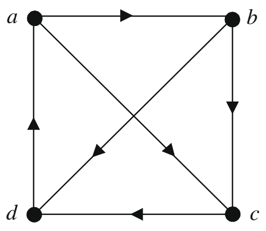

Fig. 1. The majority tournament T_0 associated with the profile Π_0 of the example.

$$\delta(T, O) = |\{(x,\ y) \in A - B\}| = |\{(x,\ y) \in B - A\}|$$

(which shows that $\delta(T, O)$ is an integer). This distance, which owns good axiomatic properties (see [4]), measures the number of disagreements between T and the ranking defined by O. From the graph theoretic point of view, $\delta(T, O)$ can also be interpreted as the number of arcs which do not have the same orientation in T and in O, or, equivalently, as the number of arcs that must be reversed in T to obtain the linear order O.

So, Slater's problem consists in computing a linear order O^* which minimizes the distance δ from T over the set $\omega(X)$:

$$\delta(T, O^*) = \min_{O \in \omega(X)} \delta(T, O),$$

or, equivalently, a linear order which minimizes the number of arcs which must be reversed in T to obtain O. This minimum number will be called the *Slater index of* T and is noted $i(T)$; an order $O^*(T)$ which minimizes the distance δ to T will be called a *Slater order of* T; a *Slater winner of* T is the first element of any Slater order of T; notice that there always exists at least one Slater winner.

The aim of this paper is to study the algorithmic complexity of Slater's problem and some of its variants. Section 2 defines these variants. The complexity results are proved in Section 3, and summarized in the conclusion, in Section 4.

2. Slater's problems

From the previous considerations, we may define several problems.

PROBLEM (P_1). Given a tournament T, compute the value of the Slater index $i(T)$ of T.
PROBLEM (P_2). Given a tournament T, compute a Slater order $O^*(T)$ of T.
PROBLEM (P_3). Given a tournament T, compute all the Slater orders $O^*(T)$ of T.
PROBLEM (P_4). Given a tournament T, compute a Slater winner of T.
PROBLEM (P_5). Given a tournament T, compute all the Slater winners of T.
PROBLEM (P_6). Given a tournament T and a vertex v of T, determine whether v is a Slater winner of T.
PROBLEM (P_7). Given a tournament $T = (X, A)$ and a linear order O defined on X, determine whether O is a Slater order of T.

The question is to know where to locate these problems inside the polynomial hierarchy (for the theory of NP-hardness, see for instance [5,13,18]; see also [15] for a comprehensive survey of the main complexity classes and [1] for a catalogue of about 500 complexity classes). In the next section, we show that Problems (P_1)–(P_6) are NP-hard while Problem (P_7) belongs to co-NP. From a practical point of view, remember that NP-hardness implies that we do not know polynomial algorithm to solve these problems (and, if P and NP are different, such algorithms do not exist): the CPU time required to solve these problems exactly can grow exponentially with respect to the size of the instance (or, here, with respect to n, the number of vertices of T) and so may soon become prohibitive. It may seem obvious that these problems share the same complexity. Indeed, there are some simple relations between some of these problems. But we must be careful. For instance, for the solution suggested by Banks (see above), it is NP-complete to determine whether a given vertex of a tournament is a Banks winner [25], but computing a Banks winner can be done in polynomial time [16].

To establish the NP-hardness of Problems (P_1)–(P_6), we shall use another problem, of which the decision version is usually called *Feedback Arc Set Problem*, here considered for tournaments:

PROBLEM (P_8). Given a tournament $T = (X, A)$, determine a subset C of A with minimum cardinality and such that the graph $T = (X, A - C)$ is without circuit.

3. Complexity results for Slater's problems

The complexity results of this section all come from the NP-hardness of (P_8). This problem, when extended to any directed graph (i.e., T is not necessarily a tournament), has been known to be NP-hard for a long time (see [19]). Recently, Alon [2], Charbit et al. [7], and Conitzer [12] proved independently that (P_8) is NP-hard. More precisely, they proved that the decision problem, called FAST below, associated with problem (P_8), is NP-complete:

Name: *Feedback Arc Set for Tournaments* (FAST)
Instance: a tournament $T = (X, A)$; an integer h;
Question: does there exist $B \subset A$ with $|B| \leqslant h$ and such that $(X, A - B)$ is without circuit?

We first study the complexity of problem (P_1) (computation of the Slater index of a tournament). For this, we define an extra notation. If U denotes a set of arcs, $\overline{U}$ will denote the set of the arcs obtained by reversing the arcs of $U : \overline{U} = \{(x, y)$ with $(y, x) \in U\}$. The decision problem associated with (P_1) is the following problem:

Name: Slater index (SI)
Instance: a tournament $T_S = (X_S, A_S)$; an integer h_S;
Question: does there exist $B_S \subset A_S$ with $|B_S| \leqslant h_S$ and such that $(X_S, (A_S - B_S) \cup \overline{B}_S)$ is a linear order?

To show that SI is NP-complete, we are going to use the NP-completeness of the problem FAST.

Theorem 1. *The problem SI is NP-complete.*

Proof. Notice first that SI belongs to NP (the proof, quite simple, is left to the reader; it is based on the fact that the size of any instance of SI is $\Theta(n^2 + \log_2 h_S)$, or also $\Theta(n^2)$, since h_S can trivially be upper-bounded by n^2).

Now, let us transform FAST into SI. For this, consider any instance (T, h) of FAST and consider it as an instance of SI: in other words, we adopt the identity for the transformation. Such a transformation is obviously polynomial.

Assume that the answer for the instance (T, h) of FAST is "yes". Then there exists $B \subset A$ with $|B| \leqslant h$ and such that $(X, A - B)$ is without any circuit. As $(X, A - B)$ is without any circuit, it can be extended into a linear order $O = (X, C)$ with $A - B \subseteq C$. Let $D = C - (A - B)$ be the set of the arcs that are added to transform $(X, A - B)$ into O. Notice that D and B are not necessarily disjoint: it may happen that we remove some arcs of T which are put again to obtain O. But it is easy to reach such a disjunction. Indeed, set $B_S = B - (B \cap D)$; in other words, B_S is the set of the arcs which are really removed from T to obtain O. As O is complete, every removed arc (x, y) of B_S is replaced by (y, x) in O. So B_S is the set of the arcs which are reversed from T to obtain the linear order $O : O = (X, (A - B_S) \cup \overline{B}_S)$. As moreover we obviously have $B_S \subset A$ and $|B_S| \leqslant |B| \leqslant h$, this involves that the instance (T, h) of SI admits the answer "yes".

Conversely, assume that the instance (T, h) of SI admits the answer "yes": there exists B_S with $|B_S| \leqslant h$ and such that $(X, (A - B_S) \cup \overline{B}_S)$ is a linear order. Set $(B = B_S$. We have obviously the required two properties: $|B| \leqslant h$ and $(X, A - B)$ is without circuit (since it is a subgraph of $(X, (A - B_S) \cup \overline{B}_S)$, which does not contain any circuit). So, the instance (T, h) of FAST admits the answer "yes".

Thus the proposed transformation keeps the answer. As it is polynomial, as FAST is NP-complete, and as SI belongs to NP, then SI is also NP-complete. □

Corollary 2. *Problem (P_1) is NP-hard.*

To study the complexity of some of the other problems, the next lemmas will be useful.

Lemma 3. *A vertex x is a Slater winner of T if and only if we have $i(T) = d^-(x) + i(T')$, where $d^-(x)$ denotes the in-degree of x in T, and where T' denotes the subtournament of T obtained from T by removing x.*

Proof. Let O be any linear order belonging to $\omega(X)$ with x as its first element. Let O' be the linear order obtained from O by removing x. Remember that the in-degree $d^-(x)$ of x in T is the number of arcs with x as their tails in T. To build O from T, it is necessary and sufficient to reverse all the arcs with x as their tails in T and then to build O' by reversing the appropriate arcs of T'. Thus the relation: $\delta(T, O) = d^-(x) + \delta(T', O')$. When O varies inside the set ω_x of the linear orders with x as their first elements, we see that O' varies inside the set $\omega(X - \{x_1\})$ of the linear orders defined on the same set of vertices as T'. Hence the following equalities hold when x is a Slater winner of T:

$$i(T) = \min_{O \in \omega_x} \delta(T,\ O) = d^-(x) + \min_{O' \in \omega(X - \{x_1\})} \delta(T',\ O') = d^-(x) + i(T').$$

Conversely, if we have $i(T)' = d^-(x) + i(T')$ for some vertex x, then x is a Slater winner of T. Indeed, any order obtained by the concatenation of x with any Slater order of T' provides a linear order at distance $d^-(x) + i(T')$ from T; if the equality $i(T) = d^-(x) + i(T')$ holds, then this order is a Slater order of T by definition. □

Lemma 4. *Let x be a Slater winner of T and let O' be a Slater order of T', where T' denotes the subtournament of T obtained from T by removing x. Then $x > O'$ is a Slater order of T.*

Proof. Let O denote the order $x > O'$. By Lemma 3, we have: $i(T) = d^-(x) + i(T')$, where $d^-(x)$ denotes the in-degree of x in T, since x is a Slater winner of T. Moreover, we have the equalities $\delta(T, O) = d^-(x) + \delta(T', O') = d^-(x) + i(T') = i(T)$: hence the result. □

Lemma 5. *A linear order $O = x_1 > x_2 > \cdots > x_n$ is a Slater order of T if and only if, for any j with $1 \leqslant j \leqslant n$, x_j is a Slater winner of the subtournament induced by $x_j, x_{j+1}, \ldots, x_n$.*

Proof. Assume that O is a Slater order of T. For any index j with $1 \leqslant j \leqslant n$, let T_j be the subtournament induced by $x_j, x_{j+1}, \ldots, x_n$. Assume that, for some j, x_j is not a Slater winner of T_j, and so that $x_j > x_{j+1} > \cdots > x_n$ is not a Slater order of T_j. Then replace $x_j > x_{j+1} > \cdots > x_n$ in O by a Slater order O_j of T_j. This provides a new order O' obtained as the concatenation of $x_1 > x_2 > \cdots > x_{j-1}$ with $O_j : O' = x_1 > \cdots > x_{j-1} > O_j$. To obtain O or O' from T, we reverse the same arcs except some arcs belonging to T_j. More precisely, we have: $\delta(T, O) - \delta(T, O') = \delta(T_j, x_j > x_{j+1} > \cdots > x_n) - \delta(T_j, O_j)$. This quantity is greater than 0 since O_j is a Slater order of T_j while $x_j > x_{j+1} > \cdots > x_n$ is supposed not to be. But in this case, O' is closer to T than O, a contradiction. So x_j is a Slater winner of this subtournament.

Conversely, assume that, for any j with $1 \leqslant j \leqslant n, x_j$ is a Slater winner of T_j. An obvious induction allows us to conclude: still with T_j denoting the subtournament induced by $x_j, x_{j+1}, \ldots, x_n$, we have that x_n is obviously a Slater order of T_n; by Lemma 4, this implies that $x_{n-1} > x_n$ is a Slater order of T_{n-1}; in its turn, still by Lemma 4, this implies that $x_{n-2} > x_{n-1} > x_n$ is a Slater order of T_{n-2}; and so on, until $O = x_1 > x_2 > \cdots > x_n$ which is a Slater order of T. □

We are going to show now that some problems are NP-hard. Remember that, for two problems (P) and (Q), we write $(P) <_T (Q)$ if (P) can be reduced to (Q) by a Turing transformation, i.e. if the existence of an algorithm $\mathscr{A}$ allowing to solve (Q) implies the possibility to solve (P) by applying $\mathscr{A}$ a polynomial number of times. Remember that, if we have $(P) <_T (Q)$ while (P) is NP-hard, then (Q) is also NP-hard. More generally, the relation $(P) <_T (Q)$ can be interpreted as: "(Q) is at least as difficult as (P)".

In the sequel, for any tournament $T, |T|$ will denote the size of T according to a "reasonable" coding, for example the number n^2 of bits required to encode the adjacency matrix of T in a binary coding.

We now prove that Problems (P_2)–(P_6) are NP-hard:

Theorem 6. *Problems (P_2)–(P_6) are NP-hard.*

Proof. To show that Problems (P_2)–(P_6) are NP-hard, let us apply appropriate Turing transformations. We consider each one of the problems.

1. NP-hardness of (P_2) (computation of a Slater order)

   Clearly, (P_2) is at least as difficult as (P_1): any algorithm $\mathscr{A}_2$ providing a Slater order O of a tournament T can be used to compute $i(T)$ with only one call to $\mathscr{A}_2$, since the computation of the distance between T and O is polynomial. Hence the relation: $(P_1) <_T (P_2)$; the NP-hardness of (P_1) implies the one of (P_2).

2. NP-hardness of (P_3) (computation of all the Slater orders)

   Knowing an algorithm to compute all the Slater orders obviously allows, with just one call, to obtain one of them. Hence the relation: $(P_2) <_T (P_3)$; the NP-hardness of (P_2) implies the one of (P_3).

3. NP-hardness of (P_4) (computation of a Slater winner)

   Assume that we know an algorithm $\mathscr{A}_4$ allowing to solve (P_4), i.e. allowing to compute a Slater winner of any tournament T. By applying $\mathscr{A}_4 n$ times, we could successively compute a Slater winner x_1 of T, then a Slater winner x_2 of $T - \{x_1\}$, then a Slater winner x_3 of $T - \{x_1, x_2\}$, and so on. So, thanks to Lemma 5, we can, by applying $\mathscr{A}_4 n$ times, compute a Slater order of T. Thus: $(P_2) <_T (P_4)$; the NP-hardness of (P_2) implies the one of (P_4).

4. NP-hardness of (P_5) (computation of all the Slater winners)

   Knowing an algorithm to compute all the Slater winners obviously allows, with just one call, to obtain one of them. Hence the relation: $(P_4) <_T (P_5)$; the NP-hardness of (P_4) implies the one of (P_5).

5. NP-hardness of (P_6) (determine whether a given vertex is a Slater winner)

   Knowing an algorithm $\mathscr{A}_6$ to determine whether a given vertex is a Slater winner allows us, by considering successively the n possible vertices of the tournament, to compute a Slater winner by applying $\mathscr{A}_6$ at most n times. Hence $(P_5) <_T (P_6)$; the NP-hardness of (P_5) implies the one of (P_6).

   This completes the proof of Theorem 6. □

Notice that the problem (P_6), which is a decision problem, is not known to belong to NP nor to co-NP. If the conjecture stated at the end of the section about (P_6) is true and if the usual conjectures of the complexity theory are true, then indeed (P_6) would not belong to NP $\cup$ co-NP.

Corollary 2 and Theorem 6 show that problems (P_1)–(P_6) are at least as difficult as any problem belonging to NP. With this respect, they provide, in a way, a lower bound of the complexities of these problems. We now try to locate them inside the polynomial hierarchy by providing upper bounds of their complexities. To state the following results, let us recall first some notation. As done by D.S. Johnson in [18], we will distinguish between decision problems (i.e. problems for which a question is set of which the answer is "yes" or "no"; note that the problems considered now does not necessarily belong to NP) and the other types of problems (as optimization problems or search problems). The class $\mathrm{P^{NP}}$ or P(NP), or Δ_2^p (or simply Δ_2), or still $\mathrm{P}^{\mathrm{NP}[n^{O(1)}]} = \bigcup_{k \geq 0} \mathrm{P}^{\mathrm{NP}[n^k]}$ contains the decision problems which can be solved by applying, with a polynomial (with respect to the size n of the instance) number of calls, a subprogram able to solve an appropriate problem belonging to NP (usually, an NP-complete problem). In other words, $\mathrm{P^{NP}}$ contains the decision problems (P) such that there exists a problem (Q) belonging to NP with $(P) <_T (Q)$, where $<_T$ still denotes the Turing transformation. Such a problem (P) is sometimes called *NP-easy* (though it can be NP-hard as well; a problem which is simultaneously NP-easy and NP-hard is said to be *NP-equivalent*: broadly speaking, this means that the complexity of an NP-equivalent problem is the same, up to some polynomials, as the complexity of NP-complete problems). This class is usually considered as the first step of the polynomial hierarchy above NP and co-NP (with this respect, the notation Δ_2 is more usual when dealing with this polynomial hierarchy; anyway, we shall keep the notation $\mathrm{P^{NP}}$, more informative and of which the meaning is easier to memorize). Indeed, $\mathrm{P^{NP}}$ contains NP obviously as well as the class co-NP: NP $\cup$ co-NP $\subseteq \mathrm{P^{NP}}$. It also contains the class $\mathrm{L^{NP}}$, also denoted by $\mathrm{P^{NP[log]}}$ or $\mathrm{P}^{\mathrm{NP}[\log n]}$, or Θ_2^p, or still $\mathrm{P}_{\|}^{\mathrm{NP}}$ or also $\mathrm{P}^{\|\mathrm{NP}}$, which contains the decision problems that can be solved by applying, a logarithmic (still with respect to the size n of the instance) number of times, a subprogram able to solve an appropriate problem belonging to NP (usually, an NP-complete problem). This class contains the classes NP and co-NP and is contained in the class $\mathrm{P^{NP}}$: NP $\cup$ co-NP $\subseteq \mathrm{L^{NP}} \subseteq \mathrm{P^{NP}}$. For the problems which are not decision problems (sometimes called "function problems"), we generalize these classes by adding "F" in front of their names (see [18]). For example, the class $\mathrm{FP^{NP}}$ or $\mathrm{F}\Delta_2^p$ (respectively the class $\mathrm{FL^{NP}}$) contains the optimization problems and the search problems which can be solved by the application of a subprogram able to solve an appropriate problem belonging to NP a polynomial (respectively logarithmic) number of times.

Theorem 7. *The problem (P_1) belongs to $\mathrm{FL^{NP}}$ and the problem (P_6) belongs to $\mathrm{L^{NP}}$.*

Proof. Let us show how an algorithm solving the NP-complete problem called SI above can be used to solve (P_1) with a number of calls upper-bounded by a logarithm in the size of the instance. Consider any instance T of (P_1). Let $\mathscr{A}$ be an algorithm to solve SI: applied to any instance (T, h) of SI, $\mathscr{A}$ indicates whether there exists a linear order at distance h or less from T. For $h = n(n-1)/2$, the answer provided by $\mathscr{A}$ is obviously "yes". Thanks to a usual dichotomous process from this initial value, we may compute the Slater index $i(T)$ of T with a number of calls to $\mathscr{A}$ which is upper-bounded by about $\log_2(n(n-1)/2)$, i.e. O(log n). Hence the result: $(P_1) \in \mathrm{FL^{NP}}$, since the size of a tournament on n vertices is about n^2.

Now, consider any instance (T, v) of (P_6), where T denotes the considered tournament (on n vertices) and v denotes a possible Slater winner of T. Let $\mathscr{A}$ still be an algorithm to solve SI. We have just seen how $\mathscr{A}$ can be exploited to compute $i(T)$ with O(log n) calls to $\mathscr{A}$. Let T_v be the tournament obtained from T by removing v. By Lemma 3, v is a Slater winner of T if and only if we have

$i(T) = d^-(v) + i(T_v)$, where $d^-(v)$ denotes the in-degree of v in T. To check whether v is a Slater winner of T thanks to $\mathscr{A}$, it suffices to apply $\mathscr{A}$ dichotomously to T to compute $i(T)$, then to T_v to compute $i(T_v)$, and last to compare $i(T) - i(T_v)$ to $d^-(v)$. As mentioned above, $i(T)$ and $i(T_v)$ can be computed by O(log n) calls to $\mathscr{A}$, and the remaining operations are negligible. Finally we obtain an algorithm solving (P_6) through an algorithm (the algorithm $\mathscr{A}$) which solves a problem of NP (the problem SI) and which is performed a number of times upper-bounded by a logarithm of the size of the data. Hence the result about the complexity of (P_6). □

Similarly, it is possible to locate $(P_2), (P_4)$ and (P_5) inside $\mathrm{FP}^{\mathrm{NP}}$, as stated in the following theorem. But, since a tournament can admit an exponential number of Slater orders with respect to n (see [8,26]), the following analysis cannot be applied to the problem (P_3).

Theorem 8. *The problems* $(P_2), (P_4)$ *and* (P_5) *belong to* $\mathrm{FP}^{\mathrm{NP}}$.

Proof. In order to prove that (P_4) belongs to $\mathrm{FP}^{\mathrm{NP}}$, we apply the same process as for (P_6) by varying the considered vertex v as the potential winner. We consider at most n vertices to find a Slater winner. Let $\mathscr{A}$ be an algorithm solving SI. To check whether a given vertex is a Slater winner can be done in O(log n) calls of $\mathscr{A}$ (see above). Since there are at most n vertices to be checked, the determination of a Slater winner can therefore be done in $O(n \log n)$ calls of $\mathscr{A}$; hence the result for (P_4). The belonging of (P_5) to $\mathrm{FP}^{\mathrm{NP}}$ is obtained in a similar way, with the same complexity, i.e. within $O(n \log n)$ calls to $\mathscr{A}$, by testing all vertices and by verifying whether each of them is a Slater winner of T. Last, by Lemma 5, we know that $O = x_1 > x_2 > \cdots > x_n$ is a Slater order of T if and only if, for all j between 1 and n, x_j is a Slater winner of the subtournament of T induced by $\{x_j, x_{j+1}, \ldots, x_n\}$. A Slater order is thus obtained by performing n times an algorithm which determines a Slater winner of a tournament. From what has been said for (P_4), this aim can be attained by applying $O(n^2 \log n)$ times the algorithm $\mathscr{A}$ solving SI. Hence the belonging of (P_2) to $\mathrm{FP}^{\mathrm{NP}}$. □

For Problem (P_7) (determine whether a given order is a Slater order of a given tournament), it can be located inside co-NP:

Theorem 9. *Problem* (P_7) *belongs to co-NP.*

Proof. Let (T, O) be an instance of (P_7). To prove that (P_7) is in co-NP, it is sufficient to guess an order O' closer to T than O (O' exists if O is not a Slater order of T). Then we just have to compute $\delta(T, O)$ and to compare it with $\delta(T, O')$. This can trivially be performed in polynomial (and even linear) time with respect to n^2, which is, broadly speaking, the size of T. So the verification is polynomial and (P_7) belongs to co-NP. □

Because of this result, (P_7) cannot be NP-complete, except if NP=co-NP. On the other hand, Theorem 9 does not permit to locate (P_7) inside co-NP precisely. So the question remains: where is (P_7) located inside the class co-NP? Is (P_7) co-NP-complete? Let us notice that the possible polynomiality of (P_7) would not be in contradiction with the NP-hardness of the other previous problems. Indeed, even if we assume that we can determine in polynomial time whether a given order is a Slater order or not, it would not indicate how to choose it, and therefore it would not indicate for instance how to find a Slater winner. On the other hand, such a polynomiality of (P_7) would have interesting consequences. Hence, it would be possible to deduce that (P_6) belongs to NP. Indeed, in order to be convinced that a given vertex v is a Slater winner, we could guess a linear order and claim that it is a Slater order. The hypothetical polynomiality of (P_7) would allow to check such a claim in polynomial time, and thus to verify in polynomial time that the vertex v is indeed a Slater winner, which would imply that (P_6) belongs to NP.

To conclude this section, we state some conjectures:

Conjectures.

1. Problem (P_1) is $\mathrm{FL}^{\mathrm{NP}}$-complete.
2. Problems (P_2), (P_4) and (P_5) are $\mathrm{FP}^{\mathrm{NP}}$-complete.
3. Problem (P_6) is L^{NP}-complete.
4. Problem (P_7) is co-NP-complete.

4. Summary

As a conclusion, let us summarize the results obtained above.

- The computation of the Slater index of a tournament (Problem (P_1)) is an NP-hard (Corollary 2) and NP-easy (Theorem 7) problem, so it is NP-equivalent. This problem belongs to $\mathrm{FL}^{\mathrm{NP}}$ (Theorem 7). The decision problem that is associated with it (problem SI) is NP-complete (Theorem 1).
- The determination of a Slater order of a tournament (Problem (P_2)), of a Slater winner of a tournament (Problem (P_4)), or of all the Slater winners of a tournament (Problem (P_5)) are NP-hard (Theorem 6) and NP-easy (Theorem 8) problems, so they are NP-equivalent. These problems belong to $\mathrm{FP}^{\mathrm{NP}}$ (Theorem 8).
- The determination of all the Slater orders of a tournament (Problem (P_3)) is an NP-hard problem (Theorem 6).
- The verification that a given vertex is a Slater winner of a tournament (Problem (P_6)) is an NP-hard (Theorem 6) and NP-easy (Theorem 7) problem, so it is NP-equivalent. This problem belongs to L^{NP} (Theorem 7).
- The verification that a given order is a Slater order of a tournament (Problem (P_7)) belongs to co-NP (Theorem 9).

We may compare these results to the ones relative to the computation of Banks winners. As said above, the computation of one Banks winner (the equivalent of Problem (P_4)) can be done in polynomial time, but the computation of all the Banks winners (the equivalent of Problem (P_5)) is NP-hard and the verification that a given vertex is a Banks winner (the equivalent of Problem (P_6)) is NP-complete. On the other hand, the computation of a Condorcet winner, when such a winner does exist, can be done in polynomial time, as well as the linear order provided by Condorcet's procedure, when this procedure provides a linear order.

Acknowledgement

I would like to thank Lucile Belgacem-Denœud for her help in writing this text. I thank also the anonymous referees for their valuable comments.

References

[1] S. Aaronson, G. Kuperberg, Complexity Zoo. <http://www.qwiki.caltech.edu/wiki/Complexity_Zoo>.
[2] N. Alon, Ranking tournaments, SIAM Journal on Discrete Mathematics 20 (1) (2006) 137–142.
[3] J. Banks, Sophisticated voting outcomes and agenda control, Social Choice and Welfare 2 (1985) 295–306.
[4] J.-P. Barthélemy, B. Monjardet, The median procedure in cluster analysis and social choice theory, Mathematical Social Sciences 1 (1981) 235–267.
[5] J.-P. Barthélemy, G. Cohen, A. Lobstein, Algorithmic Complexity and Communication Problems, UCL Press, London, 1996.
[6] C. Berge, Graphs, North-Holland Publishing Co., Amsterdam, 1985.
[7] P. Charbit, S. Thomassé, A. Yeo, The minimum feedback arc set problem is NP-hard for tournaments, Combinatorics, Probability and Computing 16 (1) (2007) 1–4.
[8] I. Charon, O. Hudry, Slater orders and Hamiltonian paths of tournaments, Electronic Notes in Discrete Mathematics 5 (2000) 60–63.
[9] I. Charon, O. Hudry, An updated survey on the linear ordering problem for weighted or unweighted tournaments, Annals of Operations Research, in press.

[10] I. Charon, O. Hudry, F. Woirgard, Ordres médians et ordres de Slater des tournois, Mathématiques, Informatique et Sciences humaines 133 (1996) 23–56.
[11] M.J.A.N. Condorcet, Caritat (marquis de), Essai sur l'application de l'analyseà la probabilité des décisions rendues à la pluralité des voix, Paris, 1785.
[12] V. Conitzer, Computing Slater rankings using similarities among candidates, in: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06), Boston, MA, USA, 2006, pp. 613–619.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[14] G.Th. Guilbaud, Les théories de l'intérêt général et le problème logique de l'agrégation, Économie appliquée 5 (4) (1952) 501–584. and *Éléments de la théorie des jeux*, Dunod, Paris, 1968.
[15] L. Hemaspaandra, Complexity classes, in: K.H. Rosen (Ed.), Handbook of Discrete and Combinatorial Mathematics, CRC Press, Boca Raton, 2000, pp. 1085–1090.
[16] O. Hudry, A note on "Banks winners in tournaments are difficult to recognize", by G.J. Woeginger, Social Choice and Welfare, vol. 23, 2004, pp. 113–114.
[17] O. Hudry, A survey on the complexity of tournament solutions, Mathematical Social Sciences 57 (2009) 292–303.
[18] D.S. Johnson, A catalog of complexity classes, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science Vol. A: Algorithms and Complexity, Elsevier, Amsterdam, 1990, pp. 67–161.
[19] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Tatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[20] J.-F. Laslier, Tournament Solutions and Majority Voting, Springer, Berlin, Heidelberg, New York, 1997.
[21] J.W. Moon, Topics on Tournaments, Holt, Rinehart and Winston, New York, 1968.
[22] K.B. Reid, Tournaments, in: J.L. Gross, J. Yellen (Eds.), Handbook of Graph Theory, CRC Press, Boca Raton, 2004, pp. 156–184.
[23] K.B. Reid, L.W. Beineke, Tournaments, in: L.W. Beineke, R.J. Wilson (Eds.), Selected Topics in Graph Theory, Academic Press, Berlin, 1978, pp. 169–204.
[24] P. Slater, Inconsistencies in a schedule of paired comparisons, Biometrika 48 (1961) 303–312.
[25] G.J. Woeginger, Banks winners in tournaments are difficult to recognize, Social Choice and Welfare 20 (3) (2003) 523–528.
[26] F. Woirgard, Recherche et dénombrement des ordres médians des tournois, PhD Thesis, ENST, Paris, 1997.