Information Processing Letters 108 (2008) 390–393

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A note on the query complexity of the Condorcet winner problem

Ariel D. Procaccia

School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel

ARTICLE INFO

Article history:
Received 18 March 2008
Received in revised form 17 July 2008
Accepted 17 July 2008
Available online 25 July 2008
Communicated by L.A. Hemaspaandra

Keywords:
Analysis of algorithms
Graph algorithms
Theory of computation
Concrete complexity
Social choice

ABSTRACT

Given an unknown tournament over $\{1,\ldots,n\}$, we show that the query complexity of the question "Is there a vertex with outdegree $n-1$?" (known as a *Condorcet winner* in social choice theory) is exactly $2n - \lfloor\log(n)\rfloor - 2$. This stands in stark contrast to the evasiveness of this property in general digraphs.

1. Introduction

An impressive body of literature is concerned with the query complexity of properties of graphs. In the deterministic model, one is presented with an unknown directed graph (digraph) $G = \langle V, E\rangle$, $V = \{1,\ldots,n\}$, and is allowed to submit queries of the form: "Is the edge (i, j) in the graph?" An algorithm for testing a property is essentially a decision tree that at each node, given the answers to its queries so far, either submits another query or outputs an answer. The complexity of a property is the worst-case number of queries that must be asked. This simple model of computation was the subject of much interest as it gives an abstraction of graphs represented by an adjacency matrix, in addition to giving lower bounds on the complexity of problems in more sophisticated models.

A fascinating branch of research focused on evasive properties of digraphs. An evasive property is a property such that one has to ask about all the edges in the graph ($n(n-1)$ in the case of digraphs) in order to test for it. Karp conjectured that any graph property that is monotone (still holds if more edges are added) and nontrivial (holds for some but not all graphs) is evasive. Over the years, many works have provided increasingly better lower bounds on the complexity of monotone, nontrivial properties [8,7,3,4], although Karp's conjecture remains open.

A prominent example of a monotone, nontrivial graph property is the existence of a vertex with outdegree $n-1$. It is straightforward to verify that this property is evasive in digraphs. In this paper, we ask whether this is also the case in tournaments.

A tournament over $N = \{1,\ldots,n\}$ is a digraph with exactly one directed edge between every two vertices. In other words, a tournament is an orientation of a complete undirected graph. Tournaments play a major role in graph theory (see, e.g., [6]), but are of particular interest in the context of social choice theory.

In an election, the voters are usually assumed to hold linear preferences over the set of candidates. Candidate $i \in N$ is said to *dominate* candidate $j \in N$ if a majority of voters prefer i to j. This dominance relation is an irreflexive, asymmetric, and complete binary relation T on $\{1,\ldots,n\}$; iTj means that i dominates j. Such a binary relation on the set of candidates is known as a tournament. Crucially, this definition and the one given above are equivalent: there is a directed edge (i, j) in tournament T's

E-mail address: arielpro@gmail.com.

graph if and only if iTj.[1] Throughout the paper, we shall use the language of social choice theory (e.g., we shall call the vertices "candidates" and use the term "dominate"), but will employ the two representations of tournaments (as a binary relation and as a digraph) interchangeably.

In a tournament, a candidate that dominates every other candidate (that is, has an outdegree of $n-1$) is known as a *Condorcet winner*.[2] The question of the existence of a Condorcet winner has a special place in social choice theory. As early as the 18th century, the Marquis de Condorcet suggested that a candidate that dominates every other candidate—should such a candidate exist—must be the winner of the election.

Our results. We show that the complexity, in the query model, of the question "Given a tournament T, is there a Condorcet winner?" is $2n - \lfloor\log(n)\rfloor - 2$. Specifically, we give matching upper and lower bounds of $2n - \lfloor\log(n)\rfloor - 2$. Our results stand in contrast to the evasiveness of this property ("Is there a candidate with outdegree $n-1$?") in general digraphs. Note that in the context of tournaments, a query is of the form $\langle i, j\rangle$, and the answer is either iTj or jTi.

Related work. Several previous works studied the query complexity of problems in tournaments. For example, Bar-Noy and Naor [1] leveraged comparison-based sorting algorithms to find a Hamiltonian path in tournaments.

The Condorcet winner problem itself has been addressed in a different concrete model of complexity. Indeed, Conitzer and Sandholm [2] studied the communication complexity of different voting rules, and, *inter alia*, gave asymptotic bounds on the communication complexity of the Condorcet winner problem. Their lower bound implies a lower bound of $\Omega(n)$ in the query model (but obtaining such an asymptotic lower bound in our model is trivial).

2. Proof of main results

Let us introduce some terminology that will ease our exposition. In analyzing algorithms, we shall refer to the point in time when t queries have been submitted as *time* t. Given a tournament T, we shall denote by $T^{(t)}$ the subset of the binary relation T based on the queries up to time t. In other words, $T^{(t)}$ is a directed graph, with the set of vertices N, and a subset of size t of the edges of T. Informally, $T^{(t)}$ is the part of the tournament T revealed to the algorithm at time t. At a certain time t, we say that $i \in N$ is *in* if, based on the queries so far, the tournament can be completed in a way that i becomes a Condorcet winner; that is, i does not have any incoming edges in $T^{(t)}$. A candidate that is not in is *out*. Finally, we define two notions of score. The *score* of candidate i at time t, denoted $s_i^{(t)}$, is the number of candidates i beats based on the first t queries, i.e. i's outdegree in $T^{(t)}$. The *knockout score* (*k-score*) of candidate i at time t, denoted $\bar{s}_i^{(t)}$, is the number of other candidates that i *ousted*; that is, the number of candidates j such that at some time $t' < t$ when j was in the algorithm queried $\langle i, j\rangle$, and received the answer iTj.

We are now ready to present our upper bound.

Theorem 2.1. $2n - \lfloor\log(n)\rfloor - 2$ *queries are sufficient to determine whether a Condorcet winner exists in a given tournament* T.

Proof. Consider the following algorithm.

Stage 1. The algorithm constructs a binary tree with n leaves that is *almost complete*, i.e. a tree of height D, such that for all levels $d \leqslant D-1$, the number of leaves at depth d is 2^d. We also require that the tree be full, that is each node has exactly 0 or 2 children. Each leaf of the tree is labeled by a distinct candidate $i \in N$. At each step, the algorithm submits the query $\langle i, j\rangle$, where i and j are two sibling leaves in the tree. The father of i and j is labeled with the winner (i if iTj and j if jTi), and the two leaves are trimmed (so the father becomes a leaf). We continue in this way until the tree becomes a singleton labeled by some candidate i_0.[3]

Stage 2. The algorithm matches i_0 against every candidate that was not matched against i_0 in Stage 1. If i_0 prevails in every competition, the algorithm returns *true*. If i_0 loses some competition, the algorithm returns *false*.

Let us consider Stage 1 of the algorithm. Each query submitted in this stage involves two candidates that are in. Therefore, each query ousts one candidate. The number of queries submitted in this stage (the number of internal nodes in the initial tree) is $n-1$. Therefore, at time $n-1$, we have only one candidate, i_0, that is in. Since the depth of any leaf in the tree is at least $\lfloor\log(n)\rfloor$, we have that

$$\bar{s}_{i_0}^{(n-1)} = s_{i_0}^{(n-1)} \geqslant \lfloor\log(n)\rfloor.$$

Moving on to Stage 2, we observe that in each step, we match i_0 versus a candidate not previously known to be dominated by i_0. If i_0 loses in one of these competitions, it follows that i_0 is not a Condorcet winner; since in Stage 2 (at time $t \geqslant n-1$) i_0 is the only candidate that is still in, we return *false*. Otherwise, we may return *true*. At the beginning of Stage 2 we have that $s_{i_0}^{(n-1)} \geqslant \lfloor\log(n)\rfloor$, hence we conclude that at most $n-1-\lfloor\log(n)\rfloor$ queries are needed in this stage. It follows that the algorithm requires a total of at most $2n - \lfloor\log(n)\rfloor - 2$ queries. □

We presently formulate and prove our lower bound.

Theorem 2.2. $2n - \lfloor\log(n)\rfloor - 2$ *queries are necessary to determine whether a Condorcet winner exists in a given tournament* T.

[1] McGarvey's Theorem [5] states that every possible tournament is induced by a preference profile of a set of voters, given enough voters.

[2] Clearly, if a Condorcet winner exists it is unique.

[3] Such a procedure is sometimes called a *Voting tree* or a *binary tree on N*.

Proof. The theorem trivially holds for $n = 1, 2$, and is easy to verify for $n = 3$. We may therefore assume that $n \geqslant 4$.

In order to establish the lower bound, we design an adversary that answers the algorithm's queries. The adversary's strategy, given the query $\langle i, j\rangle$ at time t, is as follows. If i and j are both in at time t, answer iTj if $\bar{s}_i^{(t)} \leqslant \bar{s}_j^{(t)}$, and jTi otherwise. If i is in and j is out at time t, answer iTj. Otherwise (j is in and i is out, or both are out) answer jTi.

We shall show that $2n - \lfloor\log(n)\rfloor - 3$ queries are not sufficient. We first notice that there is always at least one candidate that is in. Indeed, this is straightforward, as when i is in and j is out, the adversary answers iTj in response to the query $\langle i, j\rangle$. The second point we wish to establish is the trickiest part of the proof: we show that, given the limitation on the number of queries, no candidate is a certain Condorcet winner.

Lemma 2.3. *Let $t^* = 2n - \lfloor\log(n)\rfloor - 3$. For all $i \in N$, $s_i^{(t^*)} < n - 1$.*

Proof. Assume for contradiction that there is some $i_0 \in N$ with score $n - 1$ at time t^*. We derive a contradiction by showing that the number of queries must be too large.

We first establish an upper bound on $m = \bar{s}_{i_0}^{(t^*)}$. The crux of the proof, indeed of the adversary's design, is that for each $j \in N$ and time t when j was ousted by i_0, we have $\bar{s}_{i_0}^{(t)} \leqslant \bar{s}_j^{(t)}$. So, the k-score of the first candidate j_1 ousted by i_0 is at least 0; the k-score of the second j_2, at least 1; the third j_3, at least 2; and so on. Inductively, j_2 ousted a candidate with k-score at least 0, j_3 ousted two candidates with k-scores at least 0 and at least 1, and so on. Importantly, it is easy to see that the ousted candidates are all distinct, as a candidate can only be ousted once. Imagine a tree with some candidate with k-score h at the root, where a candidate j is a child of the candidate i that ousted it, and the leaves are candidates with k-score 0 (once again, the candidates that appear in the tree are all distinct). We shall recursively define a function $f(h)$, which gives a lower bound on the size of the tree described above as a function of the k-score h of the candidate at the root:

$$f(h) = \sum_{l=0}^{h-1} f(l) + 1, \quad f(0) = 1.$$

We may reformulate the recursion as $f(h) = 2f(h-1)$, since

$$\begin{aligned} f(h) &= \sum_{l=0}^{h-1} f(l) + 1 = f(h-1) + \left(\sum_{l=0}^{h-2} f(l) + 1\right) \\ &= 2f(h-1); \end{aligned} \tag{1}$$

notice that (1) also holds for $h = 1$. Therefore, $f(h) = 2^h$, and in particular $f(m) = 2^m$. Now, the total number of candidates is n, so it follows that $2^m \leqslant n$. We conclude that $m \leqslant \log(n)$. Since m is an integer, it holds in particular that $m \leqslant \lfloor\log(n)\rfloor$.

Having achieved this bound on m, we count the total number of queries. Since i_0 itself ousted only m candidates, other queries involving other candidates had to oust the remaining $n - 1 - m$ candidates; this accounts for at least $n - 1 - m$ queries. In addition, in order to reach a score of $n - 1$, we must have $n - 1$ queries involving i_0. The total number of queries at time t^* is therefore at least

$$2n - 2 - m \geqslant 2n - \lfloor\log(n)\rfloor - 2 > t^*.$$

We have reached a contradiction, which proves Lemma 2.3. □

Now, we must show that at time $t^* = 2n - \lfloor\log(n)\rfloor - 3$, the adversary's answers are consistent with both a tournament that has a Condorcet winner and a tournament that does not have one (if this is true then, whatever the algorithm answers, the adversary can cause the opposite answer to be correct). Since we have noticed that there is always at least one candidate that is in, by definition $T^{(t^*)}$ can be completed in a way that allows for a Condorcet winner. In order to establish the latter claim (consistency with a tournament without a Condorcet winner), we examine three cases:

Case 1. Only one candidate $i \in N$ is in at time t^*. By Lemma 2.3, $s_i^{(t^*)} \leqslant n - 2$, so there is some $j \in N$ and some query $\langle i, j\rangle$ that was not asked by the algorithm. We let jTi, and complete the rest of the tournament arbitrarily. Since all candidates except i were already out, there is no Condorcet winner.

Case 2. Exactly two candidates $i, j \in N$ are in at time t^*. Assume without loss of generality that $s_i^{(t^*)} \geqslant s_j^{(t^*)}$. By Lemma 2.3, $s_i^{(t^*)} \leqslant n - 2$. Since it must hold that $s_i^{(t^*)} + s_j^{(t^*)} \leqslant t^* \leqslant 2n - 5$ (where the second inequality follows from $n \geqslant 4$), j's score is at most $n - 3$.[4] Therefore, there is some $k \neq i, j$ such that the query $\langle j, k\rangle$ was not asked by the algorithm. In addition, as both i and j are in, there is no edge between i and j in $T^{(t^*)}$. We let kTj and jTi, and complete the rest of the tournament arbitrarily. This clearly guarantees that there is no Condorcet winner.

Case 3. At least three candidates are in at time t^*. As in Case 2, we note that the set of candidates that are in at time t^* constitutes an independent set in $T^{(t^*)}$. We can therefore create a cycle in T on the candidates that are in, and complete the rest of the tournament arbitrarily.

The proof of Theorem 2.2 is completed. □

Acknowledgements

The author would like to thank Vince Conitzer, Felix Fischer, Nati Linial, Noam Nisan, Jeff Rosenschein, and Lirong Xia for helpful discussions. The author is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

[4] This is an important point, as if there are two candidates that are in and have score $n - 2$, one of them must beat the other, hence a Condorcet winner must exist.

References

[1] A. Bar-Noy, J. Naor, Sorting, minimal feedback sets, and Hamilton paths in tournaments, SIAM Journal on Discrete Mathematics 3 (1) (1990) 7–20.
[2] V. Conitzer, T. Sandholm, Communication complexity of common voting rules, in: Proceedings of the Sixth ACM Conference on Electronic Commerce (ACM-EC), 2005, pp. 78–87.
[3] J. Kahn, M. Saks, D. Sturtevant, A topological approach to evasiveness, Combinatorica 4 (1984) 297–306.
[4] V. King, Lower bounds on the complexity of graph properties, in: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 468–476.
[5] D.C. McGarvey, A theorem on the construction of voting paradoxes, Econometrica 21 (4) (1953) 608–610.
[6] J.W. Moon, Topics on Tournaments, Holt, Reinhart and Winston, 1968.
[7] R. Rivest, S. Vuillemin, On recognizing graph properties from adjacency matrices, Theoretical Computer Science 3 (1976) 371–384.
[8] A. Rosenberg, The time required to recognize properties of graphs: A problem, SIGACT News 5 (4) (1973) 15–16.