A Linear Time Algorithm for Lewis Carroll's Voting System

Samir Khuller*

Abstract

Lewis Carroll proposed a voting system in which the winner is the candidate, who with the fewest changes in voter's preferences becomes a Condorcet winner – a candidate who beats all other candidates in pairwise majority-rule elections. Recently, Hemaspaandra, Hemaspaandra and Rothe showed that determining the winner in Carroll's system is complete for parallel access to NP. Bartholdi, Tovey and Trick showed that when the number of candidates is constant, there is a polynomial time algorithm for computing the winner. Unfortunately, the degree of their polynomial has an exponential dependence on the number of candidates. In this paper we show that there is a linear time algorithm for a fixed number of candidates.

1 Introduction

2 A linear time Algorithm

In this section we outline the main result, which is a linear time algorithm for computing the DodgsonScore of a candidate. Suppose we wish to compute the DodgsonScore of candidate i.

Suppose that there are n voters and k candidates. Each voter provides a preference list (permutation) giving a rank ordering of each candidate. Since there are at most $k!$ distinct preference lists, we assume that for each preference list of type j, we have a count of the number of voters n_j having that preference list.

For candidate i, we compute x_c which is the number of votes that candidate i still needs to beat candidate c. (If i is ahead of c in at least half the preference lists, then this is 0.) Notice that to make i a Condorcet winner, we need to have i beat each other candidate.

The key idea is to define an integer program that will compute the Dodgson score of candidate i. The size of this integer program will depend only on k (number of candidates), and not on n (number of voters). To solve this integer program, we need to use an algorithm due to Lenstra [3].

Let list j be denoted by $L(j) = (\sigma_j^1, \ldots, \sigma_j^k)$.

For each preference list of type j, we have n_j voters with this preference list. For each such j, we create ℓ_j integer variables $Y_j^p, 1 \leq p \leq \ell_j$, where there are ℓ_j candidates "ahead" of candidate i in this preference list. The variable Y_j^p will denote the number of voters in whose lists, we want to "promote" candidate i by p steps.

*Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:samir@cs.umd.edu. Research supported by NSF CAREER Award CCR-9501355.

The total cost $C(j)$ for the list of type j will be denoted by $\sum_{p=1}^{\ell_j} p \cdot Y_j^p$.

Clearly, we require $\sum_{p=1}^{\ell_j} Y_j^p \leq n_j$.

We also require that for each candidate c, we "promote" candidate i over candidate c by at least x_c votes. Luckily, this is an easy constraint to encode as follows.

In list of type j, let candidate c be ahead of candidate i. Let $Y(j,c)$ be the number of positions by which candidate c is ahead of candidate i on list type j. (If c is preferred to i and there is no other candidate in between the two, then $Y(j,c) = 1$.) Let $L(j,c) = \sum_{p=Y(j,c)}^{\ell_j} Y_j^p$ if candidate c is ahead of candidate i on list type j and $\emptyset$ otherwise.

We require that $\sum_j L(j,c) \geq x_c$.

Our objective function is to minimize $\sum_j C(j)$.

3 Acknowledgments

I am extremely grateful to Lane Hemaspaandra for igniting my interest in this problem through a wonderful talk that he gave at Maryland. I would also like to thank Kalyani Chadha and Sudipto Guha for useful discussions.

References

[1] J. Bartholdi III, C. Tovey and M. Trick, "Voting schemes for which it can be difficult to tell who won the election", *Social Choice and Welfare*, Vol 6 (1989), pp. 157–165.

[2] E. Hemaspaandra, L. A. Hemaspaandra and J. Rother, "Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP", *JACM*, Vol 44 (6), (1997), pp. 806-825.

[3] H. W. Lenstra, Jr. "Integer programming with a fixed number of variables", *Mathematics of Operations Research*, Vol 8, (1983), pp. 538–548.