Exact Analysis of Dodgson Elections: Lewis Carroll's 1876 Voting System is Complete for Parallel Access to NP[*]

Edith Hemaspaandra,[1] [**] Lane A. Hemaspaandra,[2] [***] and Jörg Rothe[3] [†]

[1] Department of Mathematics, Le Moyne College, Syracuse, NY 13214, USA
[2] Department of Computer Science, University of Rochester, Rochester, NY 14627, USA
[3] Institut für Informatik, Friedrich-Schiller-Universität Jena, 07743 Jena, Germany

Abstract. In 1876, Lewis Carroll proposed a voting system in which the winner is the candidate who with the fewest changes in voters' preferences becomes a Condorcet winner—a candidate who beats all other candidates in pairwise majority-rule elections. Bartholdi, Tovey, and Trick provided a lower bound—NP-hardness—on the computational complexity of determining the election winner in Carroll's system. We provide a stronger lower bound and an upper bound that matches our lower bound. In particular, determining the winner in Carroll's system is complete for parallel access to NP, i.e., it is complete for Θ_2^p, for which it becomes the most natural complete problem known. It follows that determining the winner in Carroll's elections is not NP-complete unless the polynomial hierarchy collapses.

1 Introduction

The Condorcet criterion is that an election is won by any candidate who defeats all others in pairwise majority-rule elections ([Con85], see [Bla58]). The Condorcet Paradox, dating from 1785 [Con85], notes that not only is it not always the case that Condorcet winners exist but, far worse, when there are more than two candidates, pairwise majority-rule elections may yield strict cycles in the aggregate preference even if each voter has non-cyclic preferences.[4] This is a widely discussed and troubling feature of majority rule (see, e.g., the discussion in [Mue89]).

In 1876, Charles Lutwidge Dodgson—more commonly referred to today by his pen name, Lewis Carroll—proposed an election system that is inspired by the Condorcet

[*] A full version of this paper, including all proofs, can be found at http://www.cs.rochester.edu/trs as UR-CS-TR-96-640. Supported in part by grants NSF-CCR-9322513 and NSF-INT-9513368/DAAD-315-PRO-fo-ab, and a University of Rochester Bridging Fellowship.

[**] `edith@bamboo.lemoyne.edu`. Work done in part while visiting Friedrich-Schiller-Universität Jena and the University of Amsterdam.

[***] `lane@cs.rochester.edu`. Work done in part while visiting Friedrich-Schiller-Universität Jena and the University of Amsterdam.

[†] `rothe@informatik.uni-jena.de`. Work done in part while visiting Le Moyne College.

[4] The standard example is an election over candidates a, b, and c in which 1/3 of the voters have preference $\langle a < b < c\rangle$, 1/3 of the voters have preference $\langle b < c < a\rangle$, and 1/3 of the voters have preference $\langle c < a < b\rangle$. In this case, though each voter individually has well-ordered preferences, the aggregate preference of the electorate is that b trounces a, c trounces b, and a trounces c. In short, individually well-ordered preferences do not necessarily aggregate to a well-ordered societal preference.

criterion,[5] yet that sidesteps the abovementioned problem [Dod76]. In particular, a Condorcet winner is a candidate who defeats each other candidate in pairwise majority-rule elections. In Carroll's system, an election is won by the candidate who is "closest" to being a Condorcet winner. In particular, each candidate is given a score that is the smallest number of exchanges of adjacent preferences in the voters' preference orders needed to make the candidate a Condorcet winner with respect to the resulting preference orders. Whatever candidate (or candidates, in the case of a tie) has the lowest score is the winner. This system admits ties but, as each candidate is assigned an integer score, no strict-preference cycles are possible.

Bartholdi, Tovey, and Trick, in their paper "Voting Schemes for which It Can Be Difficult to Tell Who Won the Election" [BTT89], raise a difficulty regarding Carroll's election system. Though the notion of winner(s) in Carroll's election system is mathematically well-defined, Bartholdi et al. raise the issue of what the *computational complexity* is of determining who is the winner. Though most natural election schemes admit obvious polynomial-time algorithms for determining who won, in sharp contrast Bartholdi et al. prove that Carroll's election scheme has the disturbing property that it is NP-hard to determine whether a given candidate has won a given election (a problem they dub `CarrollWinner`—they use the name "Dodgson" throughout, but we treat this as if they had written the equivalent "Carroll"), and that it is NP-hard even to determine whether a given candidate has tied-or-defeated another given candidate (a problem they dub `CarrollRanking`).

Bartholdi, Tovey, and Trick's NP-hardness results establish lower bounds for the complexity of `CarrollRanking` and `CarrollWinner`. *We optimally improve their two complexity lower bounds by proving that both problems are hard for Θ_2^p, the class of problems that can be solved via parallel access to NP, and we provide matching upper bounds. Thus, we establish that both problems are Θ_2^p-complete.* Bartholdi et al. explicitly leave open the issue of whether `CarrollRanking` is NP-complete: "...Thus `CarrollRanking` is as hard as an NP-complete problem, but since we do not know whether `CarrollRanking` is in NP, we can say only that it is NP-hard" [BTT89, p. 161]. From our optimal lower bounds, it follows that neither `CarrollWinner` nor `CarrollRanking` is NP-complete unless the polynomial hierarchy collapses.

As to our proof method, in order to raise the known lower bound on the complexity of Carroll elections, we first study the ways in which feasible algorithms can control Carroll elections. In particular, we establish a series of lemmas showing how polynomial-time algorithms can control oddness and evenness of election scores, "sum" over election scores, and merge elections. These lemmas then lead to our hardness results.

We remark that it is somewhat curious finding "parallel access to NP"-complete (i.e., Θ_2^p-complete) problems that were introduced almost one hundred years before complexity theory itself existed. In addition, `CarrollWinner`, which we prove complete for this class, is extremely natural when compared with previously known complete problems for this class, essentially all of which have quite convoluted forms, e.g., asking whether a given list of boolean formulas has the property that the number of formulas in the list that are satisfiable is itself an odd number (see the discussion

[5] Carroll did not use this term. Indeed, Black has shown that Carroll "almost beyond a doubt" was unfamiliar with Condorcet's work [Bla58, p. 193–194].

in [Wag87]). In contrast, the class NP, which is contained in Θ_2^p, has countless natural complete problems. Also, we mention that Papadimitriou [Pap84] has shown that `UniqueOptimalTravelingSalesperson` is complete for P^{NP}, which contains Θ_2^p.

2 Preliminaries

In this section, we introduce some standard concepts and notations from computational complexity theory [Pap94,BC93]. NP is the class of languages solvable in nondeterministic polynomial time. The polynomial hierarchy, PH, is defined as $\mathrm{PH} = \mathrm{P} \cup \mathrm{NP} \cup \mathrm{NP}^{\mathrm{NP}} \cup \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \cup \cdots$ where, for any class $\mathcal{C}$, $\mathrm{NP}^{\mathcal{C}} = \bigcup_{C \in \mathcal{C}} \mathrm{NP}^{C}$, and NP^{C} is the class of all languages that can be accepted by some NP machine that is given a black box that in unit time answers membership queries to C. The polynomial hierarchy is said to collapse if for some k the kth term in the preceding infinite union equals the entire infinite union. Computer scientists strongly suspect that the polynomial hierarchy does not collapse, though proving (or disproving) this remains a major open research issue.

The polynomial hierarchy has a number of intermediate levels. Of particular interest to us will be the level Θ_2^p. Θ_2^p is the class of all languages that can be solved via $\mathcal{O}(\log n)$ queries to some NP set (see [Wag90]). Equivalently, and more to the point for the purposes of this paper, Θ_2^p equals the class of problems that can be solved via parallel access to NP, as explained formally below. Θ_2^p falls between the first and second levels of the polynomial hierarchy: $\mathrm{NP} \subseteq \Theta_2^p \subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{NP}^{\mathrm{NP}}$. Kadin [Kad89] has proven that if NP has a sparse Turing-complete set then the polynomial hierarchy collapses to Θ_2^p, Wagner [Wag90] has shown that the definition of Θ_2^p is extremely robust, and Jenner and Torán [JT95] have shown that the robustness of the class Θ_2^p seems to fail for its function analogs.

Problems are encoded as languages of strings over some fixed alphabet Σ having at least two letters. Σ^* denotes the set of all strings over Σ. For any string $x \in \Sigma^*$, let $|x|$ denote the length of x. For any set $A \subseteq \Sigma^*$, let $\overline{A}$ denote $\Sigma^* \setminus A$. For any set $A \subseteq \Sigma^*$, let $||A||$ denote the cardinality of A. For any multiset A, $||A||$ will denote the cardinality of A. For example, if A is the multiset containing one occurrence of the preference order $\langle w < x < y \rangle$ and seventeen occurrences of the preference order $\langle w < y < x \rangle$, then $||A|| = 18$. As is standard, for each language $A \subseteq \Sigma^*$ we use χ_A to denote the characteristic function of A, i.e., $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ if $x \notin A$. Let $\langle \cdots \rangle$ be any standard, multi-arity, easily computable, easily invertible pairing function. We will also use the notation $\langle \cdots \rangle$ to denote preference orders, e.g., $\langle w < x < y \rangle$. Which use is intended will be clear from context.

In computational complexity theory, reductions are used to relate the complexity of problems. Very informally, if A reduces to B that means that, given B, one can solve A. For any a and b such that $\leq_a^b$ is a defined reduction type, and any complexity class $\mathcal{C}$, let $\mathrm{R}_a^b(\mathcal{C})$ denote $\{L \mid (\exists C \in \mathcal{C})\,[L \leq_a^b C]\}$. We refer readers to the standard source, Ladner, Lynch, and Selman [LLS75], for definitions and discussion of the standard reductions. However, we briefly and informally present to the reader the definitions of the reductions to be used in this paper. $A \leq_m^p B$ ("A polynomial-time manyone reduces to B") if there is a polynomial-time computable function f such that $(\forall x \in \Sigma^*)\,[x \in A \iff f(x) \in B]$. $A \leq_{tt}^{p} B$ ("A polynomial-time truth-table reduces to B") if there is a polynomial-time Turing machine that, on input x, computes a query that itself consists of a list of strings and, given that the machine after writing the query is then given as its answer a list telling which of the listed strings are in B, the machine then correctly determines whether x is in A (this is not the original Ladner-Lynch-Selman definition, as we have merged their querying machine and their evaluation machine, however this formulation is common and equivalent). Since a $\leq_{tt}^{p}$-reducing machine, on a given input, asks all its questions in a *parallel* (also called *non-adaptive*) manner, the informal statement above that Θ_2^p captures the complexity of "parallel access to NP" can now be expressed formally as the claim $\Theta_2^p = \mathrm{R}_{tt}^{p}(\mathrm{NP})$, which is known to hold [KSW87,Hem89].

As has become the norm, we always use hardness to denote hardness with respect to $\leq_m^p$ reductions. That is, for any class $\mathcal{C}$ and any problem A, we say that A is *$\mathcal{C}$-hard* if $(\forall C \in \mathcal{C})[C \leq_m^p A]$. For any class $\mathcal{C}$ and any problem A, we say that A is *$\mathcal{C}$-complete* if A is $\mathcal{C}$-hard and $A \in \mathcal{C}$. Completeness results are the standard method in computational complexity theory of categorizing the complexity of a problem, as a $\mathcal{C}$-complete problem A is both in $\mathcal{C}$, and is the hardest problem in $\mathcal{C}$ (in the sense that every problem in $\mathcal{C}$ can be easily solved using A).

3 The Complexity of Carroll Elections

Lewis Carroll's voting system ([Dod76], see also [NR76,BTT89]) works as follows. Each voter has strict preferences over the candidates. Each candidate is assigned a score, namely, the smallest number of sequential *exchanges of two adjacent candidates in the voters' preference orders* (henceforward called "switches") needed to make the given candidate a Condorcet winner. We say that a candidate c *ties-or-defeats* a candidate d if the score of d is not less than that of c. (Bartholdi et al. [BTT89] use the term "defeats" to denote what we, for clarity, denote by ties-or-defeats; though the notations are different, the sets being defined by Bartholdi et al. and in this paper are identical.) A candidate c is said to win the Carroll-type election if c ties-or-defeats all other candidates. Of course, due to ties it is possible for two candidates to tie-or-defeat each other, and so it is possible for more than one candidate to be a winner of the election.

Recall that all preferences are assumed to be strict. A candidate c is a *Condorcet winner* (with respect to a given collection of voter preferences) if c defeats (i.e., is preferred by strictly more than half of the voters) each other candidate in pairwise majority-rule elections. Of course, Condorcet winners do not necessarily exist for a given set of preferences, but if a Condorcet winner does exist, it is unique.

We now return to Carroll's scoring notion to clarify what is meant by the sequential nature of the switches, and to clarify by example that one switch changes only one voter's preferences. The *(Carroll) score* of any Condorcet winner is 0. If a candidate is not a Condorcet winner, but one switch (recall that a switch is an exchange of two *adjacent* preferences in the preference order of *one* voter) would make the candidate a Condorcet winner, then the candidate has a score of 1. If a candidate does not have a score of 0 or 1, but two switches would make the candidate a Condorcet winner, then the candidate has a score of 2. Note that the two switches could both be in the same voter's preferences, or could be one in one voter's preferences and one in another voter's preferences. Note also that switches are sequential. For example, with two switches, one could change a single voter's preferences from $\langle a < b < c < d\rangle$ to $\langle c < a < b < d\rangle$, where $e < f$ will denote the preference: "f is strictly preferred to e." With two switches, one could also change a single voter's preferences from $\langle a < b < c < d\rangle$ to $\langle b < a < d < c\rangle$. With two switches (not one), one could also change two voters with initial preferences of $\langle a < b < c < d\rangle$ and $\langle a < b < c < d\rangle$ to the new preferences $\langle b < a < c < d\rangle$ and $\langle b < a < c < d\rangle$. As noted earlier in this section, Carroll scores of 3, 4, etc., are defined analogously, i.e., the Carroll score of a candidate is the smallest number of sequential switches needed to make the given candidate a Condorcet winner. (We note in passing that Carroll was before his time in more ways than one. His definition is closely related to an important concept that is now known in computer science as "edit-distance"—the minimum number of operations (from some specified set of operations) required to transform one string into another. Though Carroll's single "switch" operation is not the richer set of operations most commonly used today when doing string-to-string editing (see, e.g., [SK83]), it does form a valid basis operation for transforming between permutations, which after all are what preferences are.)

Bartholdi et al. [BTT89] define a number of decision problems related to Carroll's system. They prove that given preference lists, and a candidate, and a number k, it is NP-complete to determine whether the candidate's score is at most k in the election specified by the preference lists (they call this problem `CarrollScore`). They define the problem `CarrollRanking` to be the problem of determining, given preference lists and the names of two voters, c and d, whether c ties-or-defeats d. They prove that this problem is NP-hard. They also prove that, given a candidate and preference lists, it is NP-hard to determine whether the candidate is a winner of the election.

For the formal definitions of these three decision problems, a preference order is strict (i.e., irreflexive and antisymmetric), transitive, and complete. Since we will freely identify voters with their preference orders, and two different voters can have the same preference order, we define a set of voters as a multiset of preference orders.

We will say that $\langle C, c, V\rangle$ is a *Carroll triple* if C is a set of candidates, c is a member of C, and V is a multiset of preference orders on C. Throughout this paper, we assume that, as inputs, multisets are coded as lists, i.e., if there are m voters in the voter set then $V = \langle P_1, P_2, \ldots, P_m\rangle$, where P_i is the preference order of the ith voter. $\mathit{Score}(\langle C, c, V\rangle)$ will denote the Carroll score of c in the vote specified by C and V.

Decision Problem: `CarrollScore`

Instance: A Carroll triple $\langle C, c, V\rangle$; a positive integer k.

Question: Is $\mathit{Score}(\langle C, c, V\rangle)$, the Carroll score of candidate c in the election specified by $\langle C, V\rangle$, less than or equal to k?

Decision Problem: `CarrollRanking`

Instance: A set of candidates C; two distinguished members of C, c and d; a multiset V of preference orders on C (encoded as a list, as discussed above).

Question: Does c tie-or-defeat d in the election? That is, is $Score(\langle C, c, V\rangle) \leq Score(\langle C, d, V\rangle)$?

Decision Problem: `CarrollWinner`

Instance: A Carroll triple $\langle C, c, V\rangle$.

Question: Is c a winner of the election? That is, does c tie-or-defeat all other candidates in the election?

We now state the complexity of `CarrollRanking`.

Theorem 1. `CarrollRanking` *is* Θ_2^p*-complete.*

It follows immediately—since (a) $\Theta_2^p = \mathrm{NP} \Rightarrow \mathrm{PH} = \mathrm{NP}$, and (b) $\mathrm{R}_m^p(\mathrm{NP}) = \mathrm{NP}$—that `CarrollRanking`, though known to be NP-hard [BTT89], cannot be NP-complete unless the polynomial hierarchy collapses quite dramatically.

Corollary 2. *If* `CarrollRanking` *is* NP*-complete, then* PH = NP.

Wagner has provided a useful tool for proving Θ_2^p-hardness, and we state his result below as Lemma 3. However, to be able to exploit this tool we must explore the structure of Carroll elections. In particular, we have to learn how to control oddness and evenness of election scores, how to add election scores, and how to merge elections. We do so as Lemmas 4, 5, and 7, respectively. On our way towards establishing Theorem 1, using Lemmas 3, 4, and 5 we will first establish Θ_2^p-hardness of a special problem that is closely related to `CarrollRanking`. This result is stated as Lemma 6 below. It is not hard to prove Theorem 1 using Lemma 6 and Lemma 7. Note that Lemma 7 gives more than is needed merely to establish Theorem 1. In fact, the way this lemma is stated even suffices to provide—jointly with Lemma 6—a direct proof of the Θ_2^p-hardness of `CarrollWinner`.

Lemma 3. [Wag87] *Let A be some* NP*-complete set, and let B be any set. If there exists a polynomial-time computable function g such that, for all $k \geq 1$ and all strings $x_1, \dots, x_{2k} \in \Sigma^*$ satisfying $\chi_A(x_1) \geq \chi_A(x_2) \geq \cdots \geq \chi_A(x_{2k})$, it holds that*

$$||\{i \mid x_i \in A\}|| \text{ is odd} \iff g(x_1, \dots, x_{2k}) \in B,$$

then B is Θ_2^p-hard.

Lemma 4. *There exists an* NP*-complete set A and a polynomial-time computable function f that reduces A to* `CarrollScore` *in such a way that, for every $x \in \Sigma^*$, $f(x) = \langle\langle C, c, V\rangle, k\rangle$ is an instance of* `CarrollScore` *with an odd number of voters and (1) if $x \in A$ then $Score(\langle C, c, V\rangle) = k$, and (2) if $x \notin A$ then $Score(\langle C, c, V\rangle) = k + 1$.*

Proof of Lemma 4. Bartholdi et al. [BTT89] prove the NP-hardness of `CarrollScore` by reducing `ExactCoverByThreeSets` to it. However, their reduction doesn't have the additional properties that we need in this lemma. We will construct a reduction from the standard NP-complete problem

`ThreeDimensionalMatching` (3DM) to `CarrollScore` that *does* have the additional properties we need. Let us first give the definition of 3DM:

Decision Problem: `ThreeDimensionalMatching` (3DM)

Instance: Sets M, W, X, and Y, where $M \subseteq W \times X \times Y$ and W, X, and Y are disjoint, nonempty sets having the same number of elements.

Question: Does M contain a *matching*, i.e., a subset $M' \subseteq M$ such that $||M'|| = ||W||$ and no two elements of M' agree in any coordinate?

We now describe a polynomial-time reduction f (from 3DM to `CarrollScore`) having the desired properties. Our reduction is defined by $f(x) = f'(f''(x))$, where f' and f'' are as described below. Informally, f'' turns all inputs into a standard format (instances of 3DM having $||M|| > 1$), and f' assumes its input has this format and implements the actual reduction.

Let f'' be a polynomial-time function that has the following properties.

1. If x is not an instance of 3DM or is an instance of 3DM having $||M|| \leq 1$, then $f''(x)$ will output an instance y of 3DM for which $||M|| > 1$ and, furthermore, it will hold that $y \in \text{3DM} \iff x \in \text{3DM}$.

2. If x is an instance of 3DM having $||M|| > 1$, then $f''(x) = x$.

It is clear that such functions exist. In particular, for concreteness, let $f''(x)$ be $\langle \{(d,e,p),(d,e,p')\}, \{d,d'\}, \{e,e'\}, \{p,p'\} \rangle$ if x is not an instance of 3DM or both $x \notin \text{3DM}$ and x is an instance of 3DM having $||M|| \leq 1$; let $f''(x)$ be $\langle \{(d,e,p),(d',e',p')\}, \{d,d'\}, \{e,e'\}, \{p,p'\} \rangle$ if x is an instance of 3DM having $||M|| \leq 1$ and such that $x \in \text{3DM}$; let $f''(x)$ be x otherwise.

We now describe f'. Let x be our input. If x is not an instance of 3DM for which $||M|| > 1$ then $f'(x) = 0$; this is just for definiteness, as due to f'', the only actions of f' that matter are when the input is an instance of 3DM for which $||M|| > 1$. So, suppose $x = \langle M, W, X, Y \rangle$ is an instance of 3DM for which $||M|| > 1$. Let $q = ||W||$. Define $f'(\langle M, W, X, Y \rangle) = \langle \langle C, c, V \rangle, 3q \rangle$ as follows: Let c, s, and t be elements not in $W \cup X \cup Y$. Let $C = W \cup X \cup Y \cup \{c, s, t\}$ and let V consist of the following two subparts:

1. Voters simulating elements of M. Suppose the elements of M are enumerated as $\{(w_i, x_i, y_i) \mid 1 \leq i \leq ||M||\}$. (The w_i are not intended to be an enumeration of W. Rather, they take on values from W as specified by M. In particular, w_j may equal w_k even if $j \neq k$. The analogous comments apply to the x_i and y_i variables.) For every triple (w_i, x_i, y_i) in M, we will create a voter. If i is odd, we create the voter $\langle s < c < w_i < x_i < y_i < t < \cdots \rangle$, where the elements after t are the elements of $C \setminus \{s, c, w_i, x_i, y_i, t\}$ in arbitrary order. If i is even, we do the same, except that we exchange s and t. That is, we create the voter $\langle t < c < w_i < x_i < y_i < s < \cdots \rangle$, where the elements after s are the elements of $C \setminus \{s, c, w_i, x_i, y_i, t\}$ in arbitrary order.

2. $||M|| - 1$ voters who prefer c to all other candidates.

We will now show that f has the desired properties. It is immediately clear that f'' and f', and thus f, are polynomial-time computable. It is also clear from our construction that, for each x, $f(x)$ is an instance of `CarrollScore` having an odd number of voters since, for every instance $\langle M, W, X, Y\rangle$ of 3DM with $||M|| > 1$, $f'(\langle M, W, X, Y\rangle)$ is an instance of `CarrollScore` with $||M|| + (||M|| - 1)$ voters, and since f'' always outputs instances of this form. It remains to show that, for every instance $\langle M, W, X, Y\rangle$ of 3DM with $||M|| > 1$:

(a) if M contains a matching, then *Score*$(\langle C, c, V\rangle) = 3q$, and

(b) if M does not contain a matching, then *Score*$(\langle C, c, V\rangle) = 3q + 1$.

Note that if we prove this, it is clear that f has the properties (1) and (2) of Lemma 4, in light of the properties of f''. Note that, recalling that we may now assume that $||M|| > 1$, by construction c is preferred to s and t by more than half of the voters, and is preferred to all other candidates by $||M|| - 1$ of the $2||M|| - 1$ voters.

Now suppose that M contains a matching M'. Then $||M'|| = q$, and every element in $W \cup X \cup Y$ occurs in M'. $3q$ switches turn c into a Condorcet winner as follows. For every element $(w_i, x_i, y_i) \in M'$, switch c upwards 3 times in the voter corresponding to (w_i, x_i, y_i). For example, if i is odd, this voter changes from $\langle s < c < w_i < x_i < y_i < t < \cdots\rangle$ to $\langle s < w_i < x_i < y_i < c < t < \cdots\rangle$. Let z be an arbitrary element of $W \cup X \cup Y$. Since z occurs in M', c has gained one vote over z. Thus, c is preferred to z by $||M||$ of the $2||M|| - 1$ voters. Since z was arbitrary, c is a Condorcet winner.

On the other hand, c's Carroll score can never be less than $3q$, because to turn c into a Condorcet winner, c needs to gain one vote over z for every $z \in W \cup X \cup Y$. Since c can gain only *one* vote over *one* candidate for each switch, we need at least $3q$ switches to turn c into a Condorcet winner. This proves condition (a).

To prove condition (b), first note that there is a "trivial" way to turn c into a Condorcet winner with $3q + 1$ switches: Just switch c to the top of the preference order of the first voter. The first voter was of the form $\langle s < c < w_1 < x_1 < y_1 < t < \cdots\rangle$, where the elements after t are exactly all elements in $W \cup X \cup Y \setminus \{w_1, x_1, y_1\}$, in arbitrary order. Switching c upwards $3q + 1$ times moves c to the top of the preference order for this voter, and gains one vote for c over all candidates in $W \cup X \cup Y$, which turns c into a Condorcet winner. This shows that *Score*$(C, c, V) \leq 3q + 1$, regardless of whether M has a matching or not.

Finally, note that a Carroll score of $3q$ implies that M has a matching. As before, every switch has to involve c and an element of $W \cup X \cup Y$. (This is because c must gain a vote over $3q$ other candidates—$W \cup X \cup Y$—and so any switch involving s or t would ensure that at most $3q - 1$ switches were available for gaining against the $3q$ members of $W \cup X \cup Y$, thus ensuring failure.) Thus, for every voter, c switches at most three times to become a Condorcet winner. Since c has to gain one vote in particular over each element in Y, and to "reach" an element in Y it must hold that c first switches over the elements of W and X that due to our construction fall between it and the nearest y element (among the $||M||$ voters simulating elements of M—it is clear that if any switch involves at least one of the $||M|| - 1$ dummy voters this could never lead to a Carroll score of $3q$ for c), it must be the case that c switches upwards exactly three times for exactly q voters corresponding to elements of M. This implies that the q elements of M that correspond to these q voters form a matching, thus proving condition (b). ∎

Lemma 5. *There exists a polynomial-time computable function CarrollSum such that, for all k and for all $\langle C_1, c_1, V_1\rangle$, $\langle C_2, c_2, V_2\rangle, \ldots, \langle C_k, c_k, V_k\rangle$ satisfying $(\forall j)[\|V_j\|$ is odd], it holds that CarrollSum$(\langle \langle C_1, c_1, V_1\rangle, \langle C_2, c_2, V_2\rangle, \ldots, \langle C_k, c_k, V_k\rangle\rangle)$ is a Carroll triple having an odd number of voters and such that $\sum_j Score(\langle C_j, c_j, V_j\rangle) = Score(CarrollSum(\langle \langle C_1, c_1, V_1\rangle, \langle C_2, c_2, V_2\rangle, \ldots, \langle C_k, c_k, V_k\rangle\rangle))$.*

Lemma 3, Lemma 4, and Lemma 5 together establish the Θ_2^p-hardness of a special problem that is closely related to the problems that we are interested in, `CarrollRanking` and `CarrollWinner`. Let us define the decision problem `TwoElectionRanking (2ER)`.

Decision Problem: `TwoElectionRanking (2ER)`

Instance: A pair of Carroll triples $\langle\langle C, c, V\rangle, \langle D, d, W\rangle\rangle$ both having an odd number of voters and such that $c \neq d$.

Question: Is $Score(\langle C, c, V\rangle) \leq Score(\langle D, d, W\rangle)$?

Lemma 6. `TwoElectionRanking` *is Θ_2^p-hard.*

We note in passing that `2ER` clearly is in $\mathrm{R}_{tt}^p(\mathrm{NP})$, and so from the fact that $\Theta_2^p = \mathrm{R}_{tt}^p(\mathrm{NP})$, it is clear that `2ER` is in Θ_2^p. Thus, in light of Lemma 6, `2ER` is Θ_2^p-complete. We also note in passing that, since one can trivially rename candidates, `2ER` remains Θ_2^p-complete in the variant in which "and such that $c \neq d$" is removed from the problem's definition.

In order to make the results obtained so far applicable to `CarrollRanking` and `CarrollWinner`, we need the following lemma that tells us how to merge two elections into a single election in a controlled manner.

Lemma 7. *There exist polynomial-time computable functions Merge and Merge′ such that, for all Carroll triples $\langle C, c, V\rangle$ and $\langle D, d, W\rangle$ for which $c \neq d$ and both V and W represent odd numbers of voters, there exist $\widehat{C}$ and $\widehat{V}$ such that*

(i) *$Merge(\langle C, c, V\rangle, \langle D, d, W\rangle)$ is an instance of* `CarrollRanking` *and $Merge'(\langle C, c, V\rangle, \langle D, d, W\rangle)$ is an instance of* `CarrollWinner`*,*

(ii) *$Merge(\langle C, c, V\rangle, \langle D, d, W\rangle) = \langle \widehat{C}, c, d, \widehat{V}\rangle$ and $Merge'(\langle C, c, V\rangle, \langle D, d, W\rangle) = \langle \widehat{C}, c, \widehat{V}\rangle$,*

(iii) *$Score(\langle \widehat{C}, c, \widehat{V}\rangle) = Score(\langle C, c, V\rangle) + 1$,*

(iv) *$Score(\langle \widehat{C}, d, \widehat{V}\rangle) = Score(\langle D, d, W\rangle) + 1$, and*

(v) *for each $e \in \widehat{C} \setminus \{c, d\}$, $Score(\langle \widehat{C}, c, \widehat{V}\rangle) < Score(\langle \widehat{C}, e, \widehat{V}\rangle)$.*

The results we now have established suffice to prove both Theorem 1 above and Theorem 8 below—which states that `CarrollWinner` is Θ_2^p-complete, the main result of this paper. Full proofs of the results in this paper can be found in the full version [HHR96].[6]

Theorem 8. `CarrollWinner` *is* Θ_2^p*-complete.*

Corollary 9. *If* `CarrollWinner` *is* NP*-complete, then* PH = NP.

Acknowledgments: We are indebted to J. Banks and R. Calvert for recommending Carroll elections to us as an interesting open topic worthy of study, and for providing us with the literature on this topic. We thank D. Austen-Smith, J. Banks, R. Calvert, A. Rutten, M. Scott, and J. Seiferas for helpful conversations and suggestions. L. Hemaspaandra thanks J. Banks and R. Calvert for arranging, and J. Banks for supervising, his Bridging Fellowship at the University of Rochester's Department of Political Science, during which this project was started.

References

[BC93] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity.* Prentice Hall, 1993.

[Bla58] D. Black. *Theory of Committees and Elections.* Cambridge University Press, 1958.

[BTT89] 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*, 6:157–165, 1989.

[Con85] M. J. A. N. de Caritat, Marquis de Condorcet. *Essai sur l'Application de L'Analyse à la Probabilité des Décisions Rendues à la Pluraliste des Voix.* 1785. Facsimile reprint of original published in Paris, 1972, by the Imprimerie Royale.

[Dod76] C. Dodgson. A method of taking votes on more than two issues, 1876. Pamphlet printed by the Clarendon Press, Oxford, and headed "not yet published" (see the discussions in [MU95,Bla58], both of which reprint this paper).

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HHR96] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. Technical Report TR-640, University of Rochester, Department of Computer Science, Rochester, NY, October 1996.

[6] Bartholdi et al. [BTT89] have stated without proof that `CarrollRanking` $\leq_m^p$ `CarrollWinner`. Theorem 1 plus this assertion would yield Theorem 8. However, as we wish our proof to be complete, we have proven Theorem 8 without relying on their assertion (see [HHR96]). We note in passing that our full paper implicitly provides an indirect proof of their assertion. In particular, given that one has proven Theorem 1 and Theorem 8, the assertion follows, since it follows from the definition of Θ_2^p-completeness that all Θ_2^p-complete problems are $\leq_m^p$-interreducible.

[JT95] B. Jenner and J. Torán. Computing functions with parallel queries to NP. *Theoretical Computer Science*, 141(1–2):175–193, 1995.

[Kad89] J. Kadin. $P^{NP[\log n]}$ and sparse Turing-complete sets for NP. *Journal of Computer and System Sciences*, 39(3):282–298, 1989.

[KSW87] J. Köbler, U. Schöning, and K. Wagner. The difference and truth-table hierarchies for NP. *RAIRO Theoretical Informatics and Applications*, 21:419–435, 1987.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[MU95] I. McLean and A. Urken. *Classics of Social Choice*. University of Michigan Press, 1995.

[Mue89] D Mueller. *Public Choice II*. Cambridge University Press, 1989.

[NR76] R. Niemi and W. Riker. The choice of voting systems. *Scientific American*, 234:21–27, 1976.

[Pap84] C. Papadimitriou. On the complexity of unique solutions. *Journal of the ACM*, 31(2):392–400, 1984.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[SK83] D. Sankoff and J. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, 1983.

[Wag87] K. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

[Wag90] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.