The Complexity of Bribery in Elections

Piotr Faliszewski
Department of Computer Science
University of Rochester
Rochester, NY 14627 USA
www.cs.rochester.edu/u/pfali

Edith Hemaspaandra
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623 USA
www.cs.rit.edu/˜eh

Lane A. Hemaspaandra
Department of Computer Science
University of Rochester
Rochester, NY 14627 USA
www.cs.rochester.edu/u/lane

Abstract

We study the complexity of influencing elections through bribery: How computationally complex is it for an external actor to determine whether by a certain amount of bribing voters a specified candidate can be made the election's winner? We study this problem for election systems as varied as scoring protocols and Dodgson voting, and in a variety of settings regarding homogeneous-vs.-nonhomogeneous electorate bribability, bounded-size-vs.-arbitrary-sized candidate sets, weighted-vs.-unweighted voters, and succinct-vs.-nonsuccinct input specification. We obtain both polynomial-time bribery algorithms and proofs of the intractability of bribery, and indeed our results show that the complexity of bribery is extremely sensitive to the setting. For example, we find settings in which bribery is NP-complete but manipulation (by voters) is in P, and we find settings in which bribing weighted voters is NP-complete but bribing voters with individual bribe thresholds is in P. For the broad class of elections (including plurality, Borda, k-approval, and veto) known as scoring protocols, we prove a dichotomy result for bribery of weighted voters: We find a simple-to-evaluate condition that classifies every case as either NP-complete or in P.

Introduction

This paper studies the complexity of bribery in elections, that is, the complexity of computing whether it is possible, by modifying the preferences of a given number of voters, to make some preferred candidate a winner. Recall that an election system provides a framework for aggregating voters' preferences—ideally (though there is no truly ideal voting system (Arrow 1951)) in a way that is satisfying, attractive, and natural. Societies use elections to select their leaders, establish their laws, and decide their policies. However, practical applications of elections are not restricted to people and politics. Many parallel algorithms start by electing leaders; multi-agent systems sometimes use voting for the purpose of planning (Ephrati & Rosenschein 1993); web search engines can aggregate results using methods based on elections (Dwork *et al.* 2001).

With such a range of applications, it is not surprising that elections may have a wide range of voter-to-candidate proportions. For example, in typical presidential elections there are relatively few candidates but there may be millions of voters. In the context of the web, one may consider web pages as voting on other pages by linking to them, or may consider humans to be voting on pages at a site by the time they spend on each. In such a setting we may have both a large number of voters and a large number of candidates. On the other hand, Dwork et al. (2001) suggest designing a meta search engine that treats other search engines as voters and web pages as candidates. This yields very few voters but many candidates.

Typically, we are used to the idea that each vote is equally important. However, all the above scenarios make just as much sense in a setting in which each voter has a different voting power. For example, U.S. presidential elections are in some sense weighted (different states have different voting powers in the Electoral College); shareholders in a company have votes weighted by the number of shares they own; and search engines in the above example could be weighted by their quality. Weighted voting is a natural choice in many other settings as well.

The importance of election systems naturally inspired questions regarding their resistance to abuse, and several potential dangers were identified and studied. For example, the organizers can make attempts to *control* the outcome of the elections by procedural tricks such as adding or deleting candidates or encouraging/discouraging people from voting. Classical social choice theory is concerned with the possibility or impossibility of such procedural control. However, recently it was realized that even if control is possible, it may still be difficult to find what actions are needed to effect control, e.g., because the computational problem is NP-complete. The complexity of controlling who wins the election was first studied by Bartholdi, Tovey, and Trick (1992).

Elections are endangered not only by the organizers but also by the voters (*manipulation*), who might be tempted to vote strategically (that is, not according to their true preferences) to obtain their preferred outcome. This is not desirable as it can skew the result of the elections in a way that is arguably not in the best interest of the society. The Gibbard–Satterthwaite Theorem (Gibbard 1973; Satterthwaite 1975) shows that essentially all election systems can be manipulated. So it is important to discover for which systems manipulation is *computationally difficult* to execute. This line of research was started by Bartholdi, Tovey, and Trick (1989), and was continued by many researchers, e.g, (Conitzer & Sandholm 2002a; 2002b; 2003; Conitzer, Lang, & Sandholm 2003; Elkind & Lipmaa 2005; Hemaspaandra & Hemaspaandra 2005).

Surprisingly, nobody seems to have addressed the issue of (the complexity of) bribery, i.e., attacks where the person interested in the success of a particular candidate picks a group of voters and convinces them to vote as he or she says. Bribery seems strongly motivated from both real life and from computational agent-based settings, and shares some of the flavor of both manipulation (changing voters' (reported) preferences) and control (deciding which voters to influence). This paper initiates the study of the complexity of bribery in elections.

There are many different settings in which bribery can be studied. In the simplest one we are interested only in the least number of voters we need to bribe to make our favored candidate win. A natural extension is to consider prices for each voter. In this setting, voters are willing to change their true preferences to anything we say, but only if we can meet their price. In an even more complicated setting it is conceivable that voters would have different prices depending on how we want to affect their vote (however, it is not clear how to succinctly encode a voter's price scheme). We study only the previous two scenarios.

We classify election systems with respect to bribery by in each case seeking to either prove the complexity is low by giving a polynomial-time algorithm or argue intractability via proving the NP-completeness of discovering whether bribery can affect a given case. We obtain a broad range of results showing that the complexity of bribery depends closely on the setting. For example, for weighted plurality elections, bribery is in P but jumps to being NP-complete if voters have price tags. As another example, for approval voting the manipulation problem is easily seen to be in P, but in contrast we prove that the bribery problem is NP-complete. Yet we also prove that when the bribery cost function is made more local the complexity of approval voting falls back to P. For scoring protocols with weighted voters, we obtain simple and *complete characterizations*, via two dichotomy theorems, of the complexity of bribery for the cases both of voters with and voters without price tags.

The paper is organized as follows. In the preliminary section we describe the election systems and bribery problems we are interested in. Then we provide a detailed study of bribery in plurality elections. After that we study connections between manipulation and bribery, and obtain dichotomy results for bribery under scoring protocols. Finally, we study the case of succinctly represented elections. Due to space limits, the proofs are omitted except for some brief sketches. All details can be found in the full version (Faliszewski, Hemaspaandra, & Hemaspaandra 2006).

Preliminaries

We can describe elections by providing a set $C = \{c_1, \ldots, c_m\}$ of candidates, a set V of voters specified by their preferences, and a rule for selecting winners. A voter v's preferences are represented as a list $c_{i_1} > c_{i_2} > \ldots > c_{i_m}$, $\{i_1, i_2, \ldots, i_m\} = \{1, 2, \ldots, m\}$, where c_{i_1} is the most preferred candidate and c_{i_m} is the most despised one. We assume that preferences are transitive, complete (for every two candidates each voter knows which one he or she prefers), and strict.

Social choice theory provides many election systems. Let us briefly describe those that we analyze in this paper. Winners of *plurality* elections are the candidate(s) who are the top choice of the largest number of voters (of course, these will be different voters for different winners). In *approval* voting each voter selects candidates he approves of; the candidate(s) with the most approvals win.

A *scoring protocol* for m candidates is described by a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ of nonnegative integers such that $\alpha_1 \geq \alpha_2 \ldots \geq \alpha_m$. Each time a candidate appears in the i'th position of a voter's preference list, that candidate gets α_i points; the candidate(s) who receive the most points win. Well-known examples of scoring protocols include the Borda count, plurality, k-approval, and veto voting systems, where for m-candidate elections Borda uses $\alpha = (m-1, m-2, \ldots, 0)$, plurality uses $\alpha = (1, 0, \ldots, 0, 0)$, k-approval uses $(1^k, 0^{m-k})$, and veto uses $\alpha = (1, 1, \ldots, 1, 0)$.

A Condorcet winner is a candidate who (strictly) beats all other candidates in pairwise contests, that is, a Condorcet winner beats everyone else in pairwise plurality elections. Clearly, there can be at most one Condorcet winner, but sometimes there are none. There are many voting systems that choose the Condorcet winner if one exists and use some compatible rule otherwise. One such system is that of Dodgson, where a winner is the person(s) who can become a Condorcet winner by a smallest number of switches in voters' preference lists. (A switch changes the order of two adjacent candidates on a list.) If a Condorcet winner exists, he or she is the unique winner of Dodgson's election. See Dodgson (1876) for details regarding Dodgson's voting rule, under which it is known that winner testing is complete for parallel access to NP (Hemaspaandra, Hemaspaandra, & Rothe 1997).

Now let us define the bribery problem for a given election system $\mathcal{E}$. All numbers are nonnegative integers and, unless otherwise specified, are represented in binary. $\mathcal{E}$-*bribery* is the following problem.

Given: A set C of candidates, a set V of voters specified via their preference lists, distinguished candidate p, and a nonnegative integer k.

Question: Is it possible to make p a winner of the $\mathcal{E}$ election by changing the preference lists of at most k voters?

We will speak both of the unweighted case (all voters are equal; in this paper that always holds unless "weighted" is in the problem name) and the weighted case (voters are weighted). Essentially all our results apply both to the case in which we want to make the preferred candidate a winner and to the case in which we want to make the preferred candidate the unique winner, and so we have not explicitly put a nonunique/unique setting into the problem names, and we focus on the nonunique case in our comments.

In the $\mathcal{E}$-\$bribery family of problems we assume that each voter has a price for changing his or her preference list. In such a case we ask not whether we can bribe at most k people, but whether we can make p a winner by spending at most k dollars. For example, the *plurality-weighted-\$bribery* problem can be described as follows.

Given: A set C of candidates. A collection V of voters specified via their preference lists $(prefs_1, \dots, prefs_m)$, their (nonnegative, integer) weights $(w_1, \dots, w_m)$, and their (nonnegative, integer) prices $(p_1, \dots, p_m)$. A distinguished candidate p, and a nonnegative integer k.

Question: Is there a set $B \subseteq \{1, \dots, m\}$ such that $\sum_{i \in B} p_i \le k$ and there is a way to bribe the voters from B in such a way that p becomes a winner?

Regarding the fact that in these models voters are assumed to vote as the bribes dictate, we stress that by using the term bribery, we do not intend to imply any moral failure on the part of bribe recipients: Bribes are simply payments.

As always, we say $A \le_m^p B$ (A many-one polynomial-time reduces to B) if there is a polynomial-time computable function f such that $x \in A \iff f(x) \in B$. We also use disjunctive truth-table reductions: $A \le_{dtt}^p B$ (A disjunctively truth-table reduces to B) if there is a polynomial-time procedure that on input x outputs a list of strings such that $x \in A$ if and only if at least one of those strings is in B. See, e.g., the work of Ladner, Lynch, and Selman (1975) for details regarding various reduction types. $\|S\|$ denotes the cardinality of set S.

Plurality

In this section we establish the complexity of bribery for plurality rule elections. The widespread use of plurality elections makes these results of particular relevance.

Not surprisingly, plurality-bribery is easy.

Theorem 1 *plurality-bribery is in P.*

To make sure our favorite candidate p wins, it is enough to keep on bribing voters of the most popular candidate to vote for p until that candidate becomes no more popular than p. We repeat this process until p becomes a winner. However, bribery within the plurality system is not always easy.

Theorem 2 *plurality-weighted-\$bribery is NP-complete, even for just two candidates.*

That is, bribery is easy in the simplest case, but if we allow voters to have prices and weights, then the problem becomes intractable. It is natural to ask which of the additional features (prices? weights?) is responsible for making the problem difficult. It turns out that neither of them is the sole reason and that only their combination yields enough power to make the problem NP-complete.

Theorem 3 *Both plurality-\$bribery and plurality-weighted-bribery are in P.*

A direct greedy algorithm, like that underpinning Theorem 1, fails to prove Theorem 3. Rather we approach Theorem 3's proof as follows. Assume that p will be capable of getting at least r votes (or in the weighted case, r vote weight), where r is some number to be specified later. If this is to make p a winner, we need to make sure that everyone else gets at most r votes. Thus we carefully choose enough cheapest (heaviest) voters of candidates that defeat p and bribe those voters to vote for p. Then we simply have to make sure that p gets at least r votes by bribing the cheapest (the heaviest) of the remaining voters. If during this process p ever becomes a winner without exceeding the budget (the bribe limit) then we know that bribery is possible. How do we pick the value of r? In the case of plurality-\$bribery, we can just run this procedure for all $\|V\|$ possible values, and accept exactly if it succeeds for at least one of them. For plurality-weighted-bribery a slightly trickier approach works. (Essentially, we need to try only $\|V\|$ values as well; we can start from $r = 1$ and always increase r so that one fewer person needs to be bribed in the first part of the above algorithm.)

Note that all of the above algorithms (in most cases, implicitly) assume that we bribe others to vote for p. This is a reasonable method of bribing if one wants p to become a winner, but it also has potential real-world downsides: The more people we bribe, the more likely it may be that the malicious attempts will be detected and will work against p. To minimize the chances of that happening we might instead bribe voters not to vote for p but for some other candidates. This way p does not get extra votes but might be able to take away enough voters from the most popular candidates to become a winner. We call this setting negative-bribery because the motivation of p is not to get votes for himself, but to take them away from others. Unlike Theorem 3, this version of the problem draws a very sharp line between the complexity of bribing weighted and priced voters.

Theorem 4 *plurality-weighted-negative-bribery is NP-complete, but plurality-negative-\$bribery is in P.*

Theorems 2 and 3 say that plurality-weighted-\$bribery is NP-complete, but any attempt to make it simpler immediately pushes it back to the realm of P. In fact, the situation is even more dramatic. In plurality-weighted-\$bribery we assume that both prices and weights are encoded in binary. However, if either the prices or the weights are encoded in unary, then the problem again becomes easy.

Theorem 5 *Both plurality-weighted-\$bribery$_{unary}$ and plurality-weighted$_{unary}$-\$bribery are in P.*

The proof of this theorem uses the fact that the algorithm we briefly alluded to earlier for plurality-weighted-bribery can be made to work for plurality-weighted-\$bribery as well, provided we are capable of choosing the cheapest group of voters of a particular candidate that together have at least a given weight. This is simply a knapsack problem in disguise. The knapsack problem is known to be NP-complete, but there are polynomial-time dynamic-programming solutions for the knapsack instances that occur in our unary encoded versions of plurality bribery. *Theorem 5 is particularly interesting because it says that plurality-weighted-\$bribery will be difficult only if we choose both weights and bribe prices to be high. But the prices are set by voters, and in many cases one could assume that there would be fairly low values for the bribe prices and so the problem would be easy.*

Another possible attack on the complexity of plurality-weighted-\$bribery would be through approximation algorithms. Since the knapsack problem has a polynomial-time approximation scheme, it is plausible that plurality-weighted-$bribery has one as well, and indeed we conjecture that that is the case. We mention in passing that although many researchers ask about average complexity of practically encountered NP-complete problems, it is typically very difficult to come up with a distribution of inputs that is both real-world realistic and simple enough to study.

Bribery versus Manipulation, and Two Dichotomy Theorems

The previous section provides a detailed discussion of the complexity of bribery for plurality voting. Its results were obtained by hand-crafting algorithms and reductions. It would be nicer if one could find tools that would let one inherit complexity results from the vast election systems literature. In this section we study relations between bribery and manipulation, and show how to obtain results using the relations we find. In the next section, we will discuss another fairly general tool to study certain types of bribery and manipulation problems.

Bribery can be viewed as manipulation where the set of manipulators is not fixed in advance; finding who to manipulate is part of the challenge. Note that to check whether bribery can be successful on a given input we can simply try all possible manipulations by k voters, where k is the number of bribes we are willing to make. This way for a fixed k we can disjunctively truth-table reduce any bribery problem to the analogous manipulation problem. In the following meta-theorem, bribery means any of our bribery problems except for $bribery, and manipulation represents the analogous manipulation problem.

Theorem 6 *For each fixed k it holds that bribery $\leq_{dtt}^{p}$ manipulation, where the bribery problem allows at most k bribes, and the manipulation problem allows manipulation by at most k voters.*

While simple, this result is still powerful enough to inherit some results from previous papers. Bartholdi, Tovey, and Trick (1989) discuss manipulations by single voters. Theorem 6 translates their results to the bribery case. In particular, this translation says that bribery for $k = 1$ is in P for plurality, Borda count and many other systems.

Can we strengthen Theorem 6 from constant-bounded bribery to general bribery? The answer is no: There are election systems for which bribery is NP-complete but manipulation is easy.

Theorem 7 *approval-bribery is NP-complete, but approval-manipulation and approval-weighted-manipulation are both in P.*

Algorithms for approval-manipulation and approval-weighted-manipulation are trivial: The manipulating group approves of just the favorite candidate. The NP-completeness result follows from a reduction from the NP-complete Exact-Cover-by-3Sets problem. Of course when the number of bribes is bounded by some fixed constant then approval-bribery can be solved in polynomial time.

We mention that bribery in approval elections is actually very easy, provided that one looks at a slightly different model. Our bribery problems allow us to completely modify the approval vector of a voter. This may, however, be too demanding since a voter might be willing to change some of his or her approval vector's entries but not to completely change his or her approval vector. In particular, in the *approval-bribery′* problem we will ask whether it is possible to make our favorite candidate p a winner by at most k entry changes in the approval vectors. These problems turn out to be easy.

Theorem 8 *approval-bribery′ and approval-$bribery′ are in P.*

Which of the above-discussed bribery models for approval is more practical depends on the setting. For example, bribery′ seems more natural when we look at the web and treat web pages as voting by linking to other pages. It certainly is easier to ask a webmaster to add/remove a link than to completely redesign the page.

Another approach to getting general bribery results is to reduce manipulation to bribery rather than (as in Theorem 6) reducing bribery to manipulation. We do so, though at the cost of reducing to the stronger $bribery model.

Theorem 9 *Let manipulation be some manipulation problem and let $bribery be the analogous $bribery problem (for the same election system). It holds that manipulation $\leq_{m}^{p}$ $bribery.*

To see this we need simply note that we can set price zero for the voters in the manipulating group and can set a positive price for the others. Setting the budget to zero finishes the reduction. Theorem 9 allows us to immediately classify the complexity of weighted $bribery for all scoring protocols.

Theorem 10 *For each scoring protocol $\alpha = (\alpha_1, \ldots, \alpha_m)$, if $\alpha_1 = \alpha_m$ then α-weighted-$bribery is in P; otherwise it is NP-complete.*

The proof draws on the dichotomy theorem of Hemaspaandra and Hemaspaandra (2005) for manipulation (see also the 2005 combined version of (Conitzer & Sandholm 2002a; Conitzer, Lang, & Sandholm 2003)) and our Theorems 2 and 9.

We now come to our central dichotomy result. Theorem 10 applies to $bribery. Is bribery also NP-complete? The following dichotomy theorem shows that the answer is "Yes, but in fewer cases."

Theorem 11 *For each scoring protocol $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, if $\alpha_2 = \alpha_m$ then α-weighted-bribery is in P; otherwise it is NP-complete.*

The core of the proof is to show NP-hardness. It would be nice to do so by reducing to our problem from the corresponding manipulation problems (which share the characterization's boundary line regarding the "α"s). This seems not to work, but we construct such a reduction that has the right properties whenever its inputs satisfy an additional condition (namely, that the weight of the lightest manipulating voter is at least double that of the heaviest nonmanipulator). So we would be done if this restriction of the manipulation problem were NP-hard. To show that, we by close examination of the manipulation-dichotomy proof of Hemaspaandra and Hemaspaandra (2005) prove that though the reduction from partition to that manipulation problem does not obey the desired condition in all its image elements, if we look at the image of only a certain restriction of the partition problem we can modify the thus-obtained elections to obey the desired conditions. Finally, we show by reduction from the (general) partition problem that the restricted partition problem used above is NP-hard. Thus through a chain of three reductions we establish the NP-hardness part of Theorem 11.

Although Theorem 11 does not formally imply Theorem 1 (a single P algorithm must work there for all candidate-set sizes and Theorem 11 speaks of one size at a time), it is clear that Theorem 11 immediately implies that veto-weighted-bribery is NP-complete even for 3 candidates. Yet the following result shows that the difficulty of bribery for veto comes purely from the weighted votes.

Theorem 12 *veto-bribery is in P.*

We in spirit obtained Theorem 11 by reducing from manipulation to bribery, for scoring protocols. Will that work in all other settings? The answer is no; we have designed a (artificial) voting system where checking manipulability even by just one voter is NP-complete, but checking bribability is easy. See the full version for details (Faliszewski, Hemaspaandra, & Hemaspaandra 2006).

Succinct Elections

So far we have discussed only nonsuccinct elections—ones where voters with the same preference lists (and weights, if voters are weighted) are given by listing them one at a time (as if given a stack of ballots). It is also very natural to consider the case where each preference list has its frequency conveyed via a count (in binary), and we will refer to this as "succinct" input. Succinct in curly braces within a name of a bribery problem will describe the fact that it holds in both cases, e.g., if we say that plurality-{succinct}-bribery is in P, we mean that both plurality-bribery and plurality-succinct-bribery are in P. (By the way, Theorem 1, by a similar but more careful algorithm than the one mentioned right after it, also holds for the succinct case.)

In this section we provide P membership results regarding succinctly represented elections with a fixed number of candidates. (Such results for the case of succinct representation immediately yield results for the nonsuccinct case.) The most useful tool here is Lenstra's (1983) extremely powerful result that the integer programming feasibility problem is in P when the number of variables is bounded. Lenstra's algorithm has a very large constant factor in its running time. To us, this is not a critical issue since we are interested in polynomial-time computability results and tools for obtaining them, rather than in actual optimized algorithms.

Using the integer programming approach we obtain polynomial-time algorithms for bribery under scoring protocols in both the succinct and the nonsuccinct cases. The same approach yields a similar result for manipulation. (The nonsuccinct case for manipulation was already obtained by Conitzer and Sandholm (2002a).)

Theorem 13 *For every scoring protocol $\alpha = (\alpha_1, \ldots, \alpha_m)$, both α-{succinct}-bribery and α-{succinct}-manipulation are in P.*

The power of the integer programming approach is not limited to the case of scoring protocols. In fact, the seminal paper of Bartholdi, Tovey, and Trick (1989) shows that applying this method to computing the Dodgson score in nonsuccinct elections with a fixed number of candidates yields a polynomial-time score algorithm (and though they did not address the issue of succinct elections, one can see that there too this method works perfectly). Applying an integer programming attack for the case of bribery is a bit more complicated, since one has both the issue of the bribes and the issue of the exchanges involved in computing the Dodgson scores. But even in this setting one can represent the question as a integer programming feasibility problem, and thus via Lenstra's algorithm we have the following result.

Theorem 14 *For each fixed number of candidates, DodgsonScore-{succinct}-bribery is in P when restricted to that number of candidates.*

By this we mean that in polynomial time we can test if a given bribe suffices to obtain or beat a given Dodgson score for our favored candidate (the Dodgson score of candidate c is the min. number of switches needed to be done in voters' preference lists to make c the Condorcet winner). Using binary search we may compute the minimum bribe needed to make our favored candidate have a given Dodgson score.

A similar result holds for Young elections. In Young elections (Young and Levenglick (1978); see Rothe, Spakowski, and Vogel (2003), which proves that the winner problem in Young elections is complete for parallel access to NP) the score of a candidate is the number of voters that need to be removed to make that candidate a Condorcet winner.

Theorem 15 *For each fixed number of candidates, YoungScore-{succinct}-bribery is in P when restricted to that number of candidates.*

Note that the two above results do not allow us to conclude, either for Dodgson or for Young elections, that the problem of bribing voters to make some candidate c a winner is in P. Nonetheless, a small change in the voting system does allow us to resolve a natural bribery-related winner problem. Note that bribes allow us to completely change a given voter's preference list—and this goes far beyond the switches allowed by Dodgson score-counting. We observe that one could define a Dodgson-like voting system based on bribes: Instead of counting how many switches we need to make a given candidate the Condorcet winner, count how many bribes (complete overwrites at unit cost of one voter's preference list) would suffice to guarantee such an outcome. We call this election system *Dodgson$'$*. By the above comments, for a fixed number of candidates computing winners of Dodgson$'$ elections can be done in polynomial time.

Theorem 16 *For each fixed number of candidates, the winner problem for succinct Dodgson$'$ elections is in P.*

Clearly, Dodgson$'$ elects the Condorcet winner whenever one exists, and so in practical settings it might be more reasonable to use Dodgson$'$ than Dodgson. Nonetheless, before doing so one should carefully study the properties of the new election system. Note that even though computing Dodgson′ winner for a fixed number of candidates is a polynomial-time procedure, this does not immediately imply that the bribery problem is easy for Dodgson′, and we conjecture that it is not. On the other hand, using integer programming, we obtained the following theorem.

Theorem 17 *For each fixed number of candidates, Kemeny-bribery is in* P *when restricted to that number of candidates.*

In brief, Kemeny's system elects each candidate who is most preferred in at least one preference order that maximizes the number of agreements with the voters' preferences, where for two candidates, a and b, two preference orders agree if they both place a ahead of b or both place b ahead of a. Our algorithm "or"s together a very large number of integer programming feasibility tests that cover the possible ways a candidate can become a Kemeny winner via bribery.

Conclusions

Our paper provides a detailed study of bribery with respect to plurality rule and provides tools and results regarding many other election systems, such as scoring protocols, approval voting, and Dodgson elections. Bribery seems as important an issue as manipulation and control; our paper addresses this gap in our knowledge about the complexity of voting systems.

One of the important contributions of this paper is pointing out, by concrete examples, that NP-completeness results may not guarantee the difficulty of the most natural problem instances. In particular, our Theorem 2 says that plurality-weighted-$bribery is NP-complete, but Theorem 5 observes that if either the weights or the prices are small enough, the problem can be solved efficiently. Another contribution of this paper is to relate manipulation and bribery, thus making result transfer from the former to the latter a reasonable line of attack—and one that is already exploited in spirit in the proof approach of our central dichotomy result.

Acknowledgments: We thank Samir Khuller for helpful discussions and ackn. the support of the AvH TransCoop program and NSF grants CCR-0311021 and CCF-0426761.

References

Arrow, K. 1951. *Social Choice and Individual Values*. John Wiley and Sons. Revised editon, 1963.

Bartholdi III, J.; Tovey, C.; and Trick, M. 1989. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare* 6:157–165.

Bartholdi III, J.; Tovey, C.; and Trick, M. 1992. How hard is it to control an election? *Mathematical and Computer Modeling* 16(8/9):27–40.

Bartholdi, III, J.; Tovey, C.; and Trick, M. 1989. The computational difficulty of manipulating an election. *Social Choice and Welfare* 6:227–241.

Conitzer, V., and Sandholm, T. 2002a. Complexity of manipulating elections with few candidates. In *Proc. of the 18th National Conf. on Art. Int.*, 314–319. AAAI Press.

Conitzer, V., and Sandholm, T. 2002b. Vote elicitation: Complexity and strategy-proofness. In *Proc. of the 18th National Conf. on Art. Int.*, 392–397. AAAI Press.

Conitzer, V., and Sandholm, T. 2003. Universal voting protocol tweaks to make manipulation hard. In *Proc. of the 18th Int. Joint Conf. on A. I.*, 781–788. Morgan Kaufmann.

Conitzer, V.; Lang, J.; and Sandholm, T. 2003. How many candidates are needed to make elections hard to manipulate? In *Proc. of the 9th Conf. on Theoretical Aspects of Rationality and Knowledge*, 201–214. ACM Press.

Dodgson, C. 1876. A method of taking votes on more than two issues. Clarendon Press, Oxford; reprinted in (McLean & Urken 1995).

Dwork, C.; Kumar, S.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceed. of the 10th Intl. World Wide Web Conf.*, 613–622. ACM Press.

Elkind, E., and Lipmaa, H. 2005. Small coalitions cannot manipulate voting. In *Proc. of the 9th Intl. Conf. on Financial Cryptography and Data Security*, 285–297. Springer-Verlag *Lecture Notes in Computer Science #3570*.

Ephrati, E., and Rosenschein, J. 1993. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 423–429. Morgan Kaufmann.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2006. The complexity of bribery in elections. Technical Report TR-895, Department of Computer Science, University of Rochester, Rochester, NY.

Gibbard, A. 1973. Manipulation of voting schemes. *Econometrica* 41(4):587–601.

Hemaspaandra, L., and Hemaspaandra, E. 2005. Dichotomy for voting systems. Technical Report TR-861, Department of Computer Science, University of Rochester.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM* 44(6):806–825.

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

Lenstra, Jr., H. 1983. Integer programming with a fixed number of variables. *Math. of Op. Res.* 8(4):538–548.

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

Rothe, J.; Spakowski, H.; and Vogel, J. 2003. Exact complexity of the winner problem for Young elections. *Theory of Computing Systems* 36(4):375–386.

Satterthwaite, M. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory* 10:187–217.

Young, H., and Levenglick, A. 1978. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics* 35(2):285–300.