Soc Choice Welfare (2008) 30:353–362
DOI 10.1007/s00355-007-0235-2

ORIGINAL PAPER

On the complexity of achieving proportional representation

Ariel D. Procaccia · Jeffrey S. Rosenschein · Aviv Zohar

Received: 3 January 2006 / Accepted: 6 March 2007 / Published online: 19 April 2007

Abstract We demonstrate that winner selection in two prominent proportional representation voting systems is a computationally intractable problem—implying that these systems are impractical when the assembly is large. On a different note, in settings where the size of the assembly is constant, we show that the problem can be solved in polynomial time.

1 Introduction

Andrew Carnegie, the American industrialist, famously remarked: "The first man gets the oyster, the second man gets the shell." Nevertheless, in many settings there may be multiple winners; the second, third, and fourth men (or women) may also savor the oyster. When elections are held for a legislature, for instance, there are often hundreds of winners. Although many single-winner voting rules can be generalized to elect multiple winners (Brams and Fishburn 2002), complications arise when multiple winners are involved—complications that are not necessarily shared by single-winner scenarios.

One of the basic properties that one expects from a multi-winner voting rule is *proportional representation* (PR). Intuitively, the requirement is that the proportional support enjoyed by different factions be accurately reflected in the election's results. In

A. D. Procaccia (✉) · J. S. Rosenschein · A. Zohar
School of Engineering and Computer Science, The Hebrew University of Jerusalem, Jerusalem 91904, Israel
e-mail: arielpro@cs.huji.ac.il

J. S. Rosenschein
e-mail: jeff@cs.huji.ac.il

A. Zohar
e-mail: avivz@cs.huji.ac.il

practice, this usually means that the percentage of votes secured by a party is roughly proportional to the number of seats it is awarded.

Monroe (1995) proposes an intriguing multi-winner voting scheme that strives towards proportional representation. In what Monroe refers to as the "pure" version of his scheme, the elected set of candidates minimizes voters' sum of "misrepresentation" values. More concrete versions of the system specify the source of these values. Monroe's scheme attempts to realize the following principle: if there are k winners, voters should be partitioned into k equally–sized groups, and each group assigned to a candidate, in a way that minimizes dissatisfaction of voters from their associated candidates. Chamberlin and Courant (1983) preceded Monroe with a similar voting scheme. The main difference is that they eschew the abovementioned principle, but attempt to achieve a comparable outcome using weighted voting.

Unfortunately, otherwise appealing theoretical voting schemes may have a fatal flaw: the problem of determining who the winners are may be computationally intractable. A problem is considered tractable (in this context) if its solution can be calculated using a number of computational steps that is polynomial in the size of the input; problems that may require exponential time are considered intractable. Indeed, for all practical purposes, it would take an absurdly long time to solve a problem that demands a number of steps that is exponential in the size of the input—when the input is large. Computational complexity theory (Papadimitriou 1994) classifies problems into complexity classes, such as $\mathcal{NP}$ (see Sect. 2 for details). The class of $\mathcal{NP}$-complete problems contains problems that, intuitively, are at least as hard as any other problem in $\mathcal{NP}$. Such problems are almost unanimously believed to be intractable.

Bartholdi et al. (1989) establish computational intractability as a major consideration against the adoption of a voting system. Specifically, they show that in the voting schemes devised by Charles Dodgson (a.k.a. Lewis Carroll, author of "Alice's Adventures in Wonderland") and Kemeny, it is $\mathcal{NP}$-hard to pinpoint the winners. The importance of such results is not at all diminished by the fact that the complexity analysis involved is worst-case. There is no arguing that a situation where the outcome of an election takes centuries to compute must be avoided at all costs—even if such occasions are relatively rare. This stands in stark contrast to work on the computational complexity of manipulating elections (Bartholdi and Orlin 1991; Bartholdi et al. 1989; Conitzer and Sandholm 2002; Procaccia and Rosenschein 2007); even if a manipulation problem is $\mathcal{NP}$-hard, it may still be the case that manipulation is often possible. $\mathcal{NP}$-hardness is thus not a strong enough guarantee of resistance to manipulation, although it can be a mighty objection against a voting system.

Potthoff and Brams (1998) show that in the Monroe and the Chamberlin–Courant voting schemes, the winners can be determined using integer programming. Although this provides a method to safely and accurately compute results of elections when the numbers involved are small, it is by no means an efficient method, as finding the solution of a general integer program is an $\mathcal{NP}$-complete problem (Papadimitriou 1981). Potthoff and Brams acknowledge this obstacle, and formulate an improved integer program for settings where the electorate is large. Nevertheless, the modified integer program is still of gargantuan size when the number of candidates is large.

In this paper we prove that in general, implementing the Monroe scheme or the Chamberlin–Courant scheme is an $\mathcal{NP}$-complete problem, i.e., it is hard to tell whether

there is a set of candidates with score lower than a given threshold. This implies that it is impossible to actually find the winners in polynomial time. The result holds even if the misrepresentation values are based on simple approval ballots.

We also consider a scenario where the number of winners is small, although the electorate and the number of candidates are large. Using a simple reduction, we show that in this setting winner selection can be accomplished efficiently.

The paper proceeds as follows. In Sect. 2 we briefly discuss computational complexity analysis. In Sect. 3 we formally define the voting schemes devised by Monroe, and by Chamberlin and Courant. In Sect. 4, we prove that winner selection is hard when the number of winners is large, and in Sect. 5, we demonstrate that the winner selection problem can be efficiently solved when the number of winners is small. Finally, our conclusions appear in Sect. 6.

2 Computational complexity

Computational complexity is a branch of the theory of computation, that deals with the resources required to solve a problem; in our context the resource is *time*, but there are other scarce resources, such as memory. Established in the second half of the twentieth century, it has since become a standard tool in many disciplines, including operations research and economics. In this section, we briefly discuss a tiny fragment of the theory. Readers are urged to consult (Sipser 1996) for an especially readable overview; a more detailed presentation can be found in (Papadimitriou 1994).

The *time complexity* of a problem is the worst-case number of computational steps required to solve the problem, as a function of the size of the input, using the best algorithm. The problems considered are often *decision problems*: questions to which the answer is either "yes" or "no". A decision problem is formally described as a set: instances that are contained in the set are "yes" instances, while those that are not are "no" instances.

Decision problems are classified into complexity classes; the two best-known are $\mathcal{P}$ and $\mathcal{NP}$. $\mathcal{P}$ is the class of all decision problems whose time complexity is polynomial in the size of the input. In order to show that a problem is in $\mathcal{P}$, one has to devise an algorithm that solves the problem in polynomial time for any input; such an algorithm is considered formally *efficient*.

$\mathcal{NP}$ is the class of decision problems whose positive solutions can be verified in polynomial time, given the right information. In other words, for any "yes" instance of an $\mathcal{NP}$ problem, there exists a *witness* that makes it possible to verify in polynomial time that the given instance is indeed a "yes" instance. Consider the following problem:

Definition 1 In the EXACT 3- COVER (X3C) problem, we are given a set $\mathcal{U}$ of n points such that n is divisible by 3, and a collection of m subsets of $\mathcal{U}$, $\mathcal{F} = \{F_1, \ldots, F_m\}$, each of cardinality 3, i.e., for all j, $|F_j| = 3$. We are asked whether it is possible to find $n/3$ (disjoint) subsets in $\mathcal{F}$ such that their union covers the entire set $\mathcal{U}$.

An instance is a specific choice of $\langle \mathcal{U}, \mathcal{F} \rangle$. If the given instance is a "yes" instance, a witness is a choice of $n/3$ subsets from $\mathcal{F}$. One can easily check in polynomial time whether the $n/3$ subsets indeed cover $\mathcal{U}$.

Some of the problems in $\mathcal{NP}$ are known as $\mathcal{NP}$-complete problems. Informally, these are problems that are at least as hard as any problem in $\mathcal{NP}$: if one $\mathcal{NP}$-complete problem can be solved efficiently, then any problem in $\mathcal{NP}$ can be solved efficiently. Known algorithms that solve $\mathcal{NP}$-complete problems precisely might require a number of computational steps that is exponential in the size of the input. Unfortunately, exponential functions tend to grow at an alarming rate. For example, consider an algorithm that executes 2^n computational steps, where n is the size of the input; when $n = 400$, this algorithm might require more than 10^{100} operations, which is significantly more than the number of particles in the known universe. Even with computers that are a 1,000 fold faster than today's best supercomputers, such a computation would run longer than the the current age of the universe. Furthermore, this is a trifle compared to the time required to solve the problem when the size of the input is 500.

To prove that a problem is $\mathcal{NP}$-complete, one must show that any problem in $\mathcal{NP}$ can be reformulated as this problem. Formally:

Definition 2 Let A and B be two sets, which are associated with decision problems. A function g from instances of A to instances of B is a *polynomial reduction* if and only if g can always be calculated in polynomial time, and for all instances x of the problem A:

$$x \in A \Leftrightarrow g(x) \in B.$$

If there exists such a function, we say that A *reduces* to B, and write $A \leq_p B$.

Given that one knows how to reduce problem A into problem B, one can solve A by translating it to B, and then solving B.

Definition 3 A problem B is $\mathcal{NP}$-complete if and only if:

1. $B \in \mathcal{NP}$.
2. $\mathcal{NP}$*-hardness*: For all $A \in \mathcal{NP}$, $A \leq_p B$.

As literally thousands of problems are currently known to be $\mathcal{NP}$-complete, the most straightforward way to show that a problem is $\mathcal{NP}$-complete is to provide a reduction from some problem already known to be $\mathcal{NP}$-complete; this is sufficient, since a composition of reductions is also a reduction.

The problem X3C is intuitively difficult: there is no obvious way to avoid checking many choices of $n/3$ subsets from $\mathcal{F}$—and there is a large number of possibilities. In this case, the intuition does not fail us; the following result is known (Garey and Johnson 1979):

Lemma 1 *X3C is $\mathcal{NP}$-complete.*

3 Proportional representation systems

Let the set of voters be V, and denote $|V| = n$; let the set of candidates be C, $|C| = m$. We use i to index the voters, and j to index the candidates. Furthermore, assume that k candidates are to be elected.

We begin by specifying Monroe's pure scheme (Monroe 1995). For each voter i and candidate j, a *misrepresentation value* μ_{ij} is known; this value characterizes the degree to which candidate j misrepresents voter i.

Let $\mathcal{S} = \{S \subseteq C : |S| = k\}$. Let $S \in \mathcal{S}$, and let $f_S : V \to S$ be a function that assigns voters to candidates in S. The misrepresentation score of voter i under f_S is $\mu_{if_S(i)}$. The total misrepresentation of assignment f_S is $\sum_{i \in V} \mu_{if_S(i)}$.

Monroe requires that f_S be restricted so that a similar number of voters is assigned to each candidate in S. In other words, each candidate in S must be assigned at least $\lfloor n/k \rfloor$ voters; we say that such an assignment is *balanced*. The misrepresentation score of S is the misrepresentation score of f_S, where $f_S : V \to S$ is the assignment with the minimal misrepresentation, subject to the above restriction. The k winners are the set $S \in \mathcal{S}$ with the lowest misrepresentation score.

Chamberlin and Courant (1983) adopt a similar approach; as before, one considers sets $S \in \mathcal{S}$ and assignments f_S. However, Chamberlin and Courant impose no restrictions on the assignments. Therefore, each set S is associated with the assignment $f_S : V \to S$ that minimizes misrepresentation among all possible assignments. In order to maintain proportionality, Chamberlin and Courant compensate by using weighted voting in the assembly: if S was the winning set of candidates, and the function f_S was the one that minimized misrepresentation, a candidate $j \in S$ is given weight $|f_S^{-1}(j)|$ in the assembly. For example, if a candidate was assigned two voters by the assignment that minimized misrepresentation, this candidate's weight in the assembly is two. In other words, candidates that were assigned many candidates (i.e., faithfully represent many voters) have more power after the assembly has been seated. However, for the purpose of winner selection, we consider the Chamberlin–Courant scheme to be a simpler version of Monroe's scheme, as the assignment f_S need not be balanced.

The misrepresentation values μ_{ij} may be naturally derived from ballots cast by the electorate. Assume voters cast ordinal ballots, i.e., each voter ranks all candidates, and denote by r_{ij} the rank of candidate j on the ballot of voter i. In this setting, it is reasonable to define $\mu_{ij} = r_{ij} - 1$. Another possibility, which Monroe recommends, is approval balloting: voters either approve or disapprove candidates, and may approve as many candidates as they wish. The misrepresentation value μ_{ij} is 0 if i approves j, and 1 otherwise.

4 Winner selection is intractable when the assembly is large

Before proceeding with a rigorous computational complexity analysis, we must formulate the computational problems that we face.

Definition 4

1. In the IMPLEMENTATION problem, we are given the set of voters V, the set of candidates C, integers $k, t \in \mathbb{N}$, and misrepresentation values $\mu_{ij} \in \mathbb{R}^+$. We are asked whether there exists a subset $S \subseteq C$ such that $|S| = k$, with misrepresentation at most t.

2. In the WINNER-SELECTION problem, we are given the set of voters V, the set of candidates C, and misrepresentation values $\mu_{ij} \in \mathbb{R}^+$. We must find a set of k candidates $S \in \mathcal{S}$ that minimizes misrepresentation.

Keeping in mind the discussion in Sect. 3, these formulations are valid with respect to both approaches mentioned: in Chamberlin and Courant's scheme the misrepresentation of a subset is the minimum over all assignments, while in Monroe's scheme it is the minimum over all balanced assignments.

Notice that the IMPLEMENTATION problem is formulated as a decision problem, while WINNER-SELECTION is formulated as an optimization problem. The two problems are strongly coupled, but one might prefer to formulate the decision version of WINNER-SELECTION as "given a set S, is it a winning set?" as in Bartholdi et al. (1989). However, we note that WINNER-SELECTION, as formulated in Definition 4, is at least as hard as IMPLEMENTATION. Indeed, if one can solve the former efficiently, one can also solve the latter. In more detail, if one can actually determine the winners, one would know that the minimal possible misrepresentation is, say, t'. This implies that any instance of IMPLEMENTATION with $t < t'$ is a "no" instance, while any instance with $t \geq t'$ is a "yes" instance. Therefore, if we show that IMPLEMENTATION is $\mathcal{NP}$-hard, this would imply that it is impossible to solve the WINNER-SELECTION problem in polynomial time, other than if $\mathcal{P} = \mathcal{NP}$ (and this is considered unlikely).

We shall presently prove that the IMPLEMENTATION problem is computationally intractable in both schemes. We strengthen the result by limiting voters' ballots to approval—misrepresentation values are either 0 or 1.

Remark 1 The IMPLEMENTATION problem in Chamberlin and Courant's Scheme is closely related to the k-MEDIAN problem. The latter problem is defined as follows: given a set F of m potential facilities, a set U of n users (or customers), a distance function $d : U \times F \to \mathbb{R}$, and an integer $k \leq m$, determine which k facilities to open so as to minimize the sum of the distances from each user to its closest open facility. The connection to our problem is self-evident when one thinks of the facilities as candidates, the users as voters, and the distances as misrepresentation values. Consequently, the $\mathcal{NP}$-hardness of the k-MEDIAN (Kariv and Hakimi 1979) entails the $\mathcal{NP}$-hardness of IMPLEMENTATION in the Chamberlin–Courant Scheme. However, we prove that this holds even when the misrepresentation values are 0 or 1, and extend the result to Monroe's scheme (where the assignments must be balanced).

Theorem 1 *The* IMPLEMENTATION *problem in the Monroe scheme and in the Chamberlin–Courant scheme is $\mathcal{NP}$-complete, even with approval ballots.*

Proof We must first show that the problem is in $\mathcal{NP}$. Given a "yes" instance $\langle V, C, \{\mu_{ij}\}, k, t\rangle$, it can be verified in polynomial time when a specific choice S of k candidates, coupled with an assignment f_S, are produced as a witness. One simply verifies that the misrepresentation of S with respect to f_S is at most t. In the Monroe scheme, it is also necessary to verify that f_S is balanced.

We must show that any problem in $\mathcal{NP}$ can be reduced to IMPLEMENTATION in the Chamberlin–Courant scheme or in the Monroe scheme, with approval ballots. It is sufficient to exhibit a polynomial reduction from some $\mathcal{NP}$-complete problem; we

use X3C. We begin by showing how to transform an instance of the latter problem to an instance of the former; this transformation is the reduction function.

We are given an instance $\langle \mathcal{U}, \mathcal{F} \rangle$ of X3C. We associate each point $i \in \mathcal{U}$ with a voter $i \in V$, and associate each set $F_j \in \mathcal{F}$ with a candidate $j \in C$. $\mu_{ij} = 0$ (candidate j is approved by voter i) if in the given instance of X3C it holds that $i \in F_j$; otherwise $\mu_{ij} = 1$. This should make the association between sets in $\mathcal{F}$ and candidates apparent: a candidate j is affiliated with the three voters i such that $\mu_{ij} = 0$. Finally, we set $k = |V|/3$. We ask whether there is $S \subset C$, such that $|S| = k$, with a misrepresentation score of exactly 0. Notice that this reduction between problems can be carried out in polynomial time.

It remains to show that an instance of X3C is a "yes" instance if and only if the transformed instance is a "yes" instance of our problem. Assume an instance of the former problem is a "yes" instance. Hence, there is a cover $\{F_{j_1}, \ldots, F_{j_k}\}$. Observe the set of candidates $S = \{j_1, \ldots, j_k\}$; each candidate j_l is approved by the three voters associated with points in F_{j_l}. These voters can be assigned to the candidate j_l; it follows that their misrepresentation score is 0. In fact, each one of the voters approves one of the candidates in S, as the union of the sets F_{j_l} covers $\mathcal{U}$, and hence the total misrepresentation of S is 0. Note that the assignment that we have derived is balanced, and thus is also valid in Monroe's scheme. Indeed, it must hold that the sets F_{j_l} are disjoint, as otherwise $|V|/3$ sets of cardinality 3 cannot cover a set of size $|V|$. It follows that exactly three voters are assigned to each candidate.

In the other direction, suppose that the transformed instance is a "yes" instance of IMPLEMENTATION in the Chamberlin–Courant scheme, or in Monroe's scheme. There must exist a set of candidates $S = \{j_1, \ldots, j_k\}$ with misrepresentation 0. By similar reasoning as before, the sets $\{F_{j_1}, \ldots, F_{j_k}\}$ associated with these candidates cover the entire set $\mathcal{U}$ (as a point that is not covered corresponds to a voter who does not approve any candidate, and so has misrepresentation 1). □

5 Winner selection is tractable when the assembly is small

Our hardness results in the previous section relied on the implicit assumption that the number of winners k grows with the number of voters and candidates. In this section, we explore the scenario where the number of winners is constant—this is an additional property of the problem which is implicitly or explicitly assumed throughout this section. In fact, we prove:

Theorem 2 *When k is constant,* WINNER-SELECTION *can be solved in time polynomial in n and m, both in the Chamberlin–Courant scheme and in Monroe's scheme.*

It is quite straightforward that WINNER-SELECTION in the Chamberlin–Courant scheme can be solved efficiently. In this scheme, the misrepresentation value for each $S \in \mathcal{S}$ can be easily calculated by assigning each voter i to $\operatorname{argmin}_{j \in S} \mu_{ij}$. Moreover, it holds that

$$|\mathcal{S}| = \binom{m}{k} = \frac{m!}{k!(m-k)!}.$$

This is a polynomial in m when k is constant. For example, if $k = 3$,

$$|\mathcal{S}| = \binom{m}{3} = \frac{m(m-1)(m-2)}{6} \leq m^3,$$

and in general $|\mathcal{S}| \leq m^k$. To conclude the point, one can compute the misrepresentation for every set in $\mathcal{S}$ using a number of operations that is polynomial in n and m—assuming k is constant.

Solving WINNER-SELECTION efficiently in Monroe's scheme is far trickier. Naturally, it still holds that $\mathcal{S}$ is polynomial in m. An algorithm that efficiently computes the misrepresentation score of every $S \in \mathcal{S}$ entails, by similar reasoning as before, an efficient method to select winners: one simply computes the misrepresentation for all subsets in $\mathcal{S}$ and minimizes.

Unfortunately, computing the misrepresentation for a given $S \in \mathcal{S}$ in Monroe's scheme is not straightforward. It can be accomplished using integer programming, but the solution suggested by Potthoff and Brams (1998) might be exponential in the number of candidates. Monroe (1995) himself mentions a simple algorithm: at first, each voter i is assigned to $\operatorname{argmin}_{j \in S} \mu_{ij}$. Then, the assignment is balanced by shifting voters that "will suffer the least increase in misrepresentation from the shift." Monroe adds that "determining the proper order of shifting is more tedious" when $k > 2$, "but the process is still straightforward."

Although this algorithm is indeed suitable when $k = 2$, a simple interpretation is not even guaranteed to terminate when $k \geq 3$. For example, let $S = \{1, 2, 3\}$, and let there be 3 voters. The misrepresentation values are: $\mu_{i1} = 1$, $\mu_{i2} = \mu_{i3} = 0$ for all i. Suppose that the initial assignment assigns two voters to candidate 2 and one voter to candidate 3. One of the voters assigned to candidate 2 must be shifted, and the total misrepresentation remains the same if a voter is shifted to candidate 3. Now candidate 3 is assigned 2 voters instead of one, and a voter is shifted back to candidate 2. This sequence of events is repeated infinitely.

We propose an efficient but somewhat more complicated method of solving the problem: formulating it as a *transportation problem*. In this problem, there are n sources and m destinations; each source has supply $a_i \in \mathbb{N}$ and each destination has demand $b_j \in \mathbb{N}$. For every i, j, there is a cost c_{ij} for transporting one unit of commodity from source i to destination j. The goal is to meet the demand of the destinations while minimizing the total cost. We also require that sources transport only non-fractional units of commodity.

In our case, given a set of k candidates S, we wish to find an optimal balanced assignment. We create a source for each voter and a destination for each candidate. The supply of each source is 1, and the demand of each destination is $\lfloor n/k \rfloor$. In addition, we set $c_{ij} = \mu_{ij}$. Finally, we add an additional destination $m + 1$ with demand $n - k\lfloor n/k \rfloor$, and set for all i: $c_{i,m+1} = \min_j \mu_{ij}$. If source i's unit of utility was transported to destination j in the solution, we say i *was transported* to j.

Lemma 2 *Let $OPT(S)$ be the cost of an optimal solution to the transportation problem induced by S, and let $\mu(S)$ be the misrepresentation of S (i.e., the misrepresentation of the minimizing balanced f_S). Then $OPT(S) = \mu(S)$.*

Proof We first show that $\mu(S) \leq OPT(S)$. Indeed, a solution to the above transportation problem induces a balanced assignment $f_S : V \to S$. For $j = 1, \ldots, m$, each voter i who was transported to j is assigned to candidate j. Finally, each voter i who was transported to $m + 1$ is assigned to candidate $\operatorname{argmin}_j \mu_{ij}$. The assignment is clearly balanced, as each candidate j is assigned at least the $\lfloor n/k \rfloor$ sources who were transported to j. Moreover, the misrepresentation of the assignment is exactly the cost of the given solution. We conclude this direction by noting that, in particular, the optimal solution to the transportation problem induces a balanced assignment with misrepresentation that is at most as large as the solution's cost.

In the other direction, we must prove that $\mu(S) \geq OPT(S)$. It is sufficient to show that any balanced assignment induces a solution to the transportation problem with smaller or equal cost. Given a balanced assignment, for each candidate j arbitrarily choose $\lfloor n/k \rfloor$ voters i who are assigned to j; in the solution to the transportation problem, transport these sources i to destination j. The total cost of these sources is equal to the associated voters' total misrepresentation under the balanced assignment. Now, simply transport all other sources to destination $m + 1$. Since the cost of each such source is $\min_j \mu_{ij}$, these sources' cost in the transportation problem is surely at most their associated voters' misrepresentation under the given assignment. □

Lemma 2 implies that in order to find a minimal balanced assignment in polynomial time, it is sufficient to create an instance of the transportation problem as described above (which obviously takes polynomial time), and solve it. It is known that the transportation problem can be solved in polynomial time, even when the supply and demand values are integers and it is not possible to transport fractional units of commodity (as in our case), using network flow algorithms (Cormen et al. 2001). Thus, the proof of Theorem 2 is completed.

6 Conclusion

The fact that the WINNER-SELECTION problem in the Monroe and the Chamberlin–Courant schemes is intractable strongly suggests that these schemes cannot be used in practical settings. In the worst-case, computing the election results could take eons! Viewed positively, this result implies that the integer programming formulation proposed by Potthoff and Brams (1998) is in a sense optimal: one cannot hope to find a polynomial-time solution, and different methods can be used to tackle integer programs with reasonable efficiency on average (although, of course, the solution might still take exponential time in the worst-case). Another important aspect of this result is that the type of ballots cast does not affect the complexity: the problem is hard even when misrepresentation values are only 0 or 1.

Our hardness results depend on the number of winners growing with the number of voters and candidates. When the number of winners is small, it is possible to select winners efficiently. This is straightforward in the Chamberlin–Courant scheme, but is more difficult in the Monroe scheme, as the misrepresentation of sets is based on balanced assignments. Our method finds a balanced assignment that minimizes misrepresentation in polynomial time. We stress that integer programming cannot be efficiently used in this setting. In fact, in the worst case the solution of an appropriate

integer program (of the type proposed by Potthoff and Brams for large electorates) could run exponentially long in the number of candidates, even if the number of winners is constant.

Acknowledgments The authors would like to thank the anonymous referees for their detailed and helpful comments, which in particular helped to greatly simplify the proofs in the paper. This work was partially supported by grant #898/05 from the Israel Science Foundation.

References

Bartholdi J, Orlin J (1991) Single transferable vote resists strategic voting. Soc Choice Welf 8:341–354
Bartholdi J, Tovey CA, Trick MA (1989) The computational difficulty of manipulating an election. Soc Choice Welf 6:227–241
Bartholdi J, Tovey CA, Trick MA (1989) Voting schemes for which it can be difficult to tell who won the election. Soc Choice Welf 6:157–165
Brams SJ, Fishburn PC (2002) Voting procedures. In: Arrow KJ, Sen AK, Suzumura K (eds). Handbook of social choice and welfare. (Chap. 4). North-Holland
Chamberlin JR, Courant PN (1983) Representative deliberations and representative decisions: Proportional representation and the Borda rule. Am Polit Sci Rev 77(3):718–733
Conitzer V, Sandholm T (2002) Complexity of manipulating elections with few candidates. In: Proceedings of the national conference on artificial intelligence. Edmonton, Canada, pp 314–319
Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. (2nd edn) MIT Press
Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of $\mathcal{NP}$-completeness. Freeman WH
Kariv O, Hakimi L (1979) An algorithmic approach to nework location problems. part ii: The p-medians. SIAM J Appl Math 37(3):539–560
Monroe BL (1995) Fully proportional representation. Am Polit Sci Rev 89(4):925–940
Papadimitriou CH (1981) On the complexity of integer programming. J Assoc Comput Mach 28(4)
Papadimitriou CH (1994) Computational Complexity. Addison Wesley
Potthoff RF, Brams SJ (1998) Proportional representation: broadening the options. J Theor Polit 10(2)
Procaccia AD, Rosenschein JS (2007) Junta distributions and the average-case complexity of manipulating elections. J Artif Intell Res 28:157–181
Sipser M (1996) Introduction to the theory of computation. Course Technology