Ann Math Artif Intell (2011) 62:7–26
DOI 10.1007/s10472-011-9255-9

Is computational complexity a barrier to manipulation?

Toby Walsh

Published online: 20 August 2011
© Springer Science+Business Media B.V. 2011

Abstract When agents are acting together, they may need a simple mechanism to decide on joint actions. One possibility is to have the agents express their preferences in the form of a ballot and use a voting rule to decide the winning action(s). Unfortunately, agents may try to manipulate such an election by mis-reporting their preferences. Fortunately, it has been shown that it is NP-hard to compute how to manipulate a number of different voting rules. However, NP-hardness only bounds the worst-case complexity. In this survey article, we summarize the evidence for and against computational complexity being a barrier to manipulation. We look both at techniques identified to increase complexity (for example, hybridizing together two or more voting rules), as well as other features that may change the computational complexity of computing a manipulation (for example, if votes are restricted to be single peaked then some of the complexity barriers fall away). We discuss recent theoretical results that consider the average case, as well as simple greedy and approximate methods. We also describe how computational "phase transitions", which have been fruitful in identifying hard instances of propositional satisfiability and other NP-hard problems, have provided insight into the hardness of manipulating voting rules in practice. Finally, we consider manipulation of other related problems like stable marriage and tournament problems.

Keywords Social choice · Manipulation · Computational complexity · Stable marriage

Mathematics Subject Classifications (2010) 91B14 · 91B12 · 68Q17

T. Walsh (✉)
NICTA and University of NSW, Sydney, Australia
e-mail: toby.walsh@nicta.com.au

1 Introduction

In multiagent systems, we often need to combine the preferences of several agents. For instance, SCATS (the Sydney Coordinated Adaptive Traffic System) is a complex distributed multiagent system used to control the traffic lights in Sydney and over 140 other cities around the world [1]. Each intersection in a SCATS systems is controlled by a separate kerbside computer. Based on the current traffic demands, each intersection has a preferred plan for the cycle time for the traffic lights, as well as the "split", the ratio of this time given to the different approaches. To ensure traffic flows smoothly, a coordination mechanism is used to coordinate the cycle times and splits of neighbouring intersections. This can ensure "green waves" of traffic long along key arterial roads. A simple, effective and well understood mechanism for aggregating together preferences of multiple agents is to apply a voting rule [2, 3]. Each agent expresses a preference ordering over the set of candidates, and an election is held to compute the winner. Going back to the SCATS example, each intersection casts a vote for its preferred plan. A new plan is then adopted when it receives a sufficient threshold of votes.

Voting is therefore a topic of interest to researchers developing multi-agent systems. Social choice studies the properties of different voting methods. However, as we shall see, multiagent systems raise some fresh issues not previously considered in social choice, mostly due to computational aspects of the problem. For example, the number of candidates running in an election in a multiagent system can be much larger than the number that typically run in human elections. In a web meta-search engine, each candidate may be one of a large number of different webpages. In addition, the agents may have significant computational resources at their disposal which they could use to decide how to vote strategically. These features can give rise to a number of interesting computational questions. For example, how does an agent compute a strategic vote which improves the outcome compared to voting truthfully? And what is the complexity of such a computation when the number of candidates can be large?

2 Voting

For elections where we need to pick between just two candidates, one of the simplest and most natural voting rules is majority voting. Each agent identifies whichever of the two candidates they prefer most, and the candidate with the most votes wins. There is little more to say about elections between two candidates. There are strong practical, theoretical and idealogical reasons to choose majority voting when there are just two candidates. It has, for instance, good axiomatic properties. Agents cannot profit by misreporting their most preferred candidate, and participating in the election can only help their most preferred candidate win. In addition, many other voting rules degenerate to majority voting when we have just two candidates.

Unfortunately, when there are more than two candidates, voting is more problematic. A number of paradoxes and impossibility results have been identified. Many different voting rules have been proposed over the years to tackle such issues. We

shall consider some of the most commonly used voting rules where n agents vote over m different candidates:

Plurality: the candidate ranked in first place by the largest number of agents wins. For example, plurality is used in many electoral systems including elections for the British Parliament. When there are only two candidates, plurality is the same as majority voting.

Scoring rules: suppose $(w_1, \ldots, w_m)$ is a vector of weights, and the ith candidate in a vote scores w_i, then the winner is the candidate with highest total score over all the votes. The *plurality* rule has the weight vector $(1, 0, \ldots, 0)$, the *veto* rule has the vector $(1, 1, \ldots, 1, 0)$, whilst the *Borda* rule has the vector $(m-1, m-2, \ldots, 0)$. The Borda rule was used to elect members of the French Academy of Sciences until this was over-turned by Napoleon. It is still used today to elect minority members of the National Assembly of Slovenia, as well as in various organizations including Robocup. Although Borda proposed the method in 1770, Llull may have invented it as early as the 13th century.

Cup: The winner is the result of a series of pairwise majority elections (or matches) between candidates. The rule requires the sequence of matches to be given. This is called the *agenda* or *voting tree*. It is an interesting open question which voting rules can be implemented by means of a voting tree. For instance, whilst any voting rule over three candidates can be implemented by a voting tree, the Copeland rule over 8 or more candidates is not implementable by any voting tree. A special case is *simple* voting trees when each candidate occurs exactly once in the leaves of the tree. We will limit ourselves to simple voting trees.

Plurality with runoff: If one candidate has a majority, they win. Otherwise everyone but the two candidates with the most votes are eliminated and the winner is chosen using the majority rule. A variation of this rule (used in the French National Assembly) is to eliminate all candidates with less than a given threshold and then run a plurality election.

Single Transferable Vote (STV): This rule requires up to $m-1$ rounds. In each round, the candidate with the least number of agents ranking them first is eliminated until one of the remaining candidates has a majority. STV is used in a number of parliamentary elections, as well as by many organizations including the General Synod of the Church of England and the Academy Awards. STV is also known as the Hare or the Hare-Clark rule. Hare is generally credited with proposing STV voting in 1857, although the idea of transferable votes can be traced further back to Hill in 1821. In 1896, Clark was responsible for the introduction of STV into parliamentary elections in Tasmania where it remains in use today.

Approval: Each agent labels candidates as approved or not. The candidate with the most number of approvals wins. Approval voting is used by many professional organizations including the Mathematical Association of America, the Institute of Management Sciences and the American Statistical Association. It is often favored when multiple candidates need to be elected.

Copeland: The candidate with the highest Copeland score wins. The Copeland score of candidate i is $\sum_{j \neq i} (N(i, j) > \frac{n}{2}) - (N(i, j) < \frac{n}{2})$ where $N(i, j)$ is the number of agents preferring i to j. The Copeland winner is the candidate that wins the most

pairwise elections. In the second order Copeland rule, if there is a tie, the winner is the candidate whose defeated competitors have the largest sum of Copeland scores. The Copeland rule is used, for example, by the World Chess Federation to decide the winner of a round robin tournament. A variant of the Copeland rule is also credited to Llull [4].

Nanson's and Baldwin's rules: These are iterated versions of the Borda rule. In Nanson's rule, we compute the Borda scores and eliminate any candidate with less than half the mean score. We repeat until there is an unique winner. In Baldwin's rule, we compute the Borda scores and eliminate the candidate with the lowest score. We again repeat until there is an unique winner. Just as STV can be seen as an iterated version of plurality voting, these two rules can be seen as iterated versions of the Borda rule. Both rules have been used in real world elections. Nanson's rule, for example, is in use in elections in the University of Adelaide, whilst Baldwin's rule was used by the Dialectic Society of Trinity College, the oldest collegiate society in Australia.

All these voting rules can also be easily modified to work with *weighted* votes. A vote of integer weight k can be viewed as k agents who vote identically.

3 Properties of voting rules

One way to study different voting rules like these is to consider the (desirable) properties that they possess or might possess. This is often referred to as an *axiomatic* approach. A large number of simple properties have been put forwards:

Anonymity: Each agent is treated the same. In particular, if we change the order of the agents then the result is unchanged.

Neutrality: Each candidate is treated the same. More precisely, if we swap around the names of the candidates then the winner is just the swap of the previous winner.

Surjectivity: Every possible result is possible. In particular, for every possible result, there exists a set of votes which gives this result.

Unanimity: The unanimous candidate wins. More precisely, if one candidate is most preferred by every agent then this candidate wins.

All the voting rules defined so far possess these properties of anonymity, neutrality, surjectivity and unanimity. However, there are other more complex properties which might be considered desirable that are not possessed by all voting rules.

Monotonicity: Increasing preferences for a candidate does not hurt the candidate. In particular, if we move a candidate up an agent's preference order then the candidate does not go from winning to losing. For example, the plurality voting rule is monotonic. By comparison, plurality with runoff is not monotonic.

Condorcet consistency: If there is a candidate who beats all others in pairwise elections then this candidate is elected [5]. The Copeland rule is Condorcet consistent. By comparison, many of the other voting rules including plurality and Borda are not Condorcet consistent.

Strategy proof: It is in the best interests of each agent to order outcomes as they actually prefer. More precisely, an agent cannot make their preferred candidate

win by mis-reporting their preferences. A voting rule that is strategy proof is also called *non-manipulable*.

Non-dictatorial: More than one agent decides the outcome. More precisely, the final result is not the same as the preferences of one particular agent.

Unfortunately, whilst each of these properties might sound reasonable on its own, it is impossible for any voting rule to have certain combinations of these properties. For example, the Gibbard-Satterthwaite theorem [6, 7] states:

Theorem 1 (Gibbard-Sattertwhaite theorem) *If there are three or more candidates, and the voting rule is strategy proof and surjective then it is dictatorial.*

Since most voting rules are surjective and non-dictatorial, it follows that they are manipulable. That is, agents can profit by voting strategically. This could be especially problematic in multiagent systems as agents may have significant computational resources at their disposal to invest in manipulating the outcome.

4 Escaping manipulation

An appealing escape from results like the Gibbard-Satterthwaite theorem was proposed by Bartholdi, Tovey and Trick in an influential paper two decades ago [8]. Perhaps it is computationally so difficult to find a successful manipulation that agents have little option but to report their true preferences? To illustrate this idea, they demonstrated that it is NP-hard to compute a manipulation of the second order Copeland rule. Shortly after, Bartholdi and Orlin proved that it is NP-hard to compute a manipulation of the more well known Single Transferable Voting (STV) rule [9]. A whole subfield of social choice has since grown from this proposal, proving that it is NP-hard to compute manipulations of various voting rules under different assumptions (e.g. [10]). Computational complexity results of this kind typically vary along several dimensions.

Weighted or unweighted votes: Are the votes weighted or unweighted? A number of real-world settings like elected assemblies and shareholder meetings use weighted votes. Weights are of interest from a computational perspective for a number of reasons. First, weights can increase the computational complexity of computing a manipulation. For example, computing how a single agent manipulates the Borda rule is polynomial with unweighted votes but NP-hard with weighted votes [10]. Second, the weighted case provides insight into the unweighted case when we have probabilistic information about how agents will vote. For instance, if it is NP-hard to compute how to manipulate an election with weighted votes, then it is NP-hard to compute the probability of a candidate winning when we have a probability distribution about how the agents will cast their unweighted votes [11].

Bounded or unbounded number of candidates: Are there a small number of candidates? Is this number fixed? Or is it allowed to grow as we increase the size of problems? For example, with unweighted votes, computing a manipulation of the STV rule is polynomial if we bound the number of candidates and only NP-hard when the number of candidates is allowed to grow with problem size [9]. Indeed,

with a bounded number of candidates and unweighted votes, it is polynomial to compute how to manipulate most voting rules [10]. With weighted votes, on the other hand, it is NP-hard to compute how to manipulate many voting rules even if we have only a small number of candidates (e.g. with Borda and weighted votes, it is NP-hard to compute how to manipulate an election with just three candidates [10]).

One manipulator or a coalition of manipulators: Is a single agent trying to manipulate the results or is there a coalition of agents acting together to manipulate the result? A single agent may be unlikely to have enough influence to be able to change the outcome of the election. A coalition, on the other hand, is more likely to be able to manipulate the result. With voting rules like STV, it is NP-hard to compute how a single agent needs to vote strategically to manipulate the result [9]. With some other voting rules, however, we need a coalition of agents of some fixed size or larger before manipulation becomes NP-hard to compute (e.g. depending on how ties are handled, it is polynomial to compute how a single agent can manipulate the result of the Copeland rule but NP-hard to compute how two agents can together manipulate the result [12]). Finally, there are many voting rules where it is only NP-hard to compute a manipulation when the number of manipulators is unbounded (e.g. it is NP-hard to compute a manipulation of the veto rule with weighted votes but polynomial with unweighted votes [10]).

Complete or incomplete information: Many NP-hardness results assume that the manipulating agents have complete information about the other votes. Of course, we may not know precisely how other agents will vote. However, there are several reasons why results that assume complete knowledge about the votes are interesting. First, if it is NP-hard to compute how to manipulate the election with complete information then it is also NP-complete when we have incomplete information. Second, the complete knowledge case informs us about the case when there is uncertainty about how agents will vote. For instance, if it is NP-hard to compute how a coalition can manipulate an election based on complete knowledge about the votes of the other agents then it is also NP-hard to compute how to manipulate an election when the coalition only has probabilistic information about the other votes [10].

Constructive or destructive manipulation: Are the manipulators trying to make a given candidate win (constructive manipulation) or prevent a given candidate from winning (destructive manipulation)? Constructive manipulations can be computationally harder to compute than destructive manipulations. For instance, computing destructive manipulation of the Copeland or veto rules by a coalition of agents with weighted votes is polynomial but computing constructive manipulation is NP-hard [10]. On the other hand, there are also voting rules where both types of manipulation have the same computational complexity (e.g. both constructive and destructive manipulation of plurality are polynomial to compute, whilst constructive and destructive manipulation of plurality with runoff for weighted votes are both NP-hard to compute [10]).

In Table 1, we give a representative selection of results about the computationally complexity of manipulating different voting rules. See [10] for references to many of these results, and [13–15] for some of the more recent. One of the last open problems in this area, the worst-case complexity of computing a manipulation of

Table 1 Computational complexity of deciding if various voting rules can be manipulated by a single agent (unweighted votes) or by a coalition of agents (weighted votes)

	Unweighted votes		Weighted votes					
	Constructive manipulation		Constructive			Destructive		
# candidates			2	3	≥4	2	3	≥4
# manipulators	1	≥2						
plurality	P	P	P	P	P	P	P	P
cup	P	P	P	P	P	P	P	P
Borda	P	NP-c	P	NP-c	NP-c	P	P	P
veto	P	P	P	NP-c	NP-c	P	P	P
STV	NP-c	NP-c	P	NP-c	NP-c	P	NP-c	NP-c
plurality with runoff	P	P	P	NP-c	NP-c	P	NP-c	NP-c
Copeland	P	P	P	P	NP-c	P	P	P
Nanson	NP-c	NP-c	P	P	NP-c	P	P	NP-c
Baldwin	NP-c	NP-c	P	NP-c	NP-c	P	NP-c	NP-c

P means that the problem is polynomial, *NP-c* that the problem is NP-complete. For example, constructive manipulation of the veto rule is polynomial for unweighted votes or for weighted votes with 2 candidates, and NP-hard for three or more. On the other hand, destructive manipulation of the veto rule is polynomial for weighted votes with 2 or more candidates

the Borda rule when votes are unweighted, was only recently settled [13, 14]. This problem is polynomial for a single manipulation but NP-hard for two manipulators. There is, however, considerable evidence that it is not hard to compute manipulations for the Borda rule in practice. Brelsford et al. have proved that weighted (and thus unweighted) Borda manipulation has a fully polynomial time approximation scheme (FPTAS) [16]. This means that a close to optimal manipulation can be found in polynomial time. Similarly Zuckerman et al. have given a simple greedy algorithm to calculate a manipulation that, in the unweighted case, uses only one more manipulator than is optimal [17]. Finally, we have proposed two greedy algorithms for computing manipulations of the Borda rue based on intuitions from the bin-packing and multiprocessor scheduling domains [18]. Although we have not been able to provide worst-case performance guarantees for these two algorithms, our empirical evaluation shows that they outperform the greedy algorithm of Zuckerman et al. and can find optimal manipulations in the vast majority of the randomly generated elections that we tested. All these results suggest that Borda voting can usually be manipulated with relative ease. They demonstrate that simple greedy methods have good characteristics and that computational complexity is limited to potentially rare worst case scenarios.

5 Other types of manipulation

Strategic voting is just one of several ways that the result of an election can be manipulated. For example, the chair person running the election may be able to manipulate the result by controlling how the election is run [19]. A number of

different types of control have been studied that might be thought of as forms of manipulation:

Candidate addition/deletion: Can the chair add (some given number of) spoiler candidates to make a particular candidate win (or lose)? By deleting (some given number of other) candidates, can the chair make a particular candidate win (or lose)?

Agent addition/deletion: Can the chair add (some given number of) new agents with given preferences in order to make a particular candidate win (or lose)? By deleting (some given number of other) agents, can the chair make a particular candidate win (or lose)?

Candidate partitioning: Can the chair partition the candidates into two sets so that a particular candidate wins (or loses) when all the candidates in the first set go into an election, and the survivors go into an election against the second set?

Run off partitioning: Can the chair partition the candidates into two sets so that a particular candidate wins (or loses) when all the candidates in each set go into an election against each other, and the survivors of both elections go forwards into a final election?

Agent partitioning: Can the chair partition the agents into two sets so that a particular candidate wins (or loses) when we run an election with each set of agents, and the survivors of these two elections go forwards into a final election in which all agents vote?

The chair can even try to manipulate the result by trying multiple methods of attack [20]. For example, the chair might simultaneously remove some agents, add other new agents, and introduce a spoiler candidate. Loosely speaking, we say that a voting rule is *immune* to a particular type of control if the chair cannot change the result. For example, with approval voting, adding spoiler candidates cannot stop a candidate being approved. On the other hand, a voting system is said to be *(computationally) vulnerable* to control if it is not immune to control and computing how the chair can control the election takes polynomial time. For instance, with plurality voting, the chair can easily calculate how many agents need to be deleted to ensure the current winner loses. The chair only needs to delete those agents who put the current winner in first place. Finally, a voting system is said to be *resistant* to control if it is not immune to control but computing what the chair needs to do is computationally intractable (i.e., NP-hard). For example, with plurality voting, it is NP-hard for the chair to compute which spoiler candidates to add to ensure a particular candidate wins. Not surprisingly, some of these types of control are related. For instance, a voting system is immune to constructive control by deleting candidates if and only if it is immune to destructive control by adding candidates.

The list we gave of different types of control is not exhaustive. There are several other important types of control that have received attention in the literature. For example, an agent may try to change the result by means of bribery [21]. In the simplest setting, we can consider the least number of agents we need to bribe to make a given candidate win. However, other more complex settings have been considered. Each agent may be willing to change their vote to whatever we choose, but only if we can meet a given price. An even more complex case is when agents have prices that differ depending on how we change their vote. For example, with unweighted votes, all three forms of bribery are polynomial to compute for plurality

but NP-hard for approval voting [21]. Besides bribery, other types of control have been studied. For instance, in a cup election, the chair may be able to manipulate the result by re-ordering the agenda [22–24]. This is closely related to the problem of finding a good (or in some sense optimal) ranking of players in a knock out tournament.

6 Tie-breaking

The manipulability of an election can depend on how ties are broken. A typical assumption made in the literature is that we have an odd number of agents as this may make ties impossible. However, other assumptions are also made. For example, we might assume that ties are broken in favour of the manipulator. Suppose the manipulator can make their preferred candidate win assuming ties are broken in their favour but ties are in fact broken at random. Then we can conclude that the manipulator can increase the chance of getting their preferred result. Tie-breaking can even introduce computational complexity into manipulation. For example, computing how to manipulate the Copeland rule with weighted votes is polynomial if ties are scored with 1 but NP-hard if they are scored with 0 [12]. Results where the computational complexity depends precisely on how ties are broken raise concerns that computational complexity will be a significant barrier to manipulation in these cases. It is likely that computationally difficult manipulation problems will be limited to the (potentially rare) cases where the election is critically hung between two outcomes. Our recent empirical results add weight to these concerns. We discuss these results in Section 10.

7 Hybrid rules

New voting rules have been proposed that are designed specifically to be difficult to manipulate. One general construction that tends to make manipulation difficult is to "hybridize" together two or more existing voting rules. For example, we might add one elimination pre-round to the election in which candidates are paired off and only the most preferred goes through [25]. This generates a new voting rule that is often computationally hard to manipulate. In fact, voting rules can now become even PSPACE-hard to manipulate. For instance, adding such a pre-round to plurality makes computing a manipulation NP-hard, #P-hard, or PSPACE-hard depending on whether the schedule of the pre-round is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively [25]. Manipulation can thus be in an even higher complexity class than NP (assuming PSPACE $\neq$ NP). Such hybrid voting rules also inherit some (but not all) of the properties of the voting rule from which they are constructed. For example, if the initial rule is Condorcet consistent, then adding a pre-round preserves Condorcet consistency.

Other types of voting rules can be hybridized together. For example, we can construct a hybrid of the Borda and STV rules in which we run two rounds of Borda, eliminating the lowest scoring candidate each time, and then apply the STV rule to the remaining candidates. Such hybrids are often resistant to manipulation.

For example, many hybrids of STV and of Borda are NP-hard to manipulate [26]. Another way to hybridize together two or more voting rules is to use some aspect of the particular election (the preferences of the agents, or the names of the candidates) to pick which voting rule is used to compute the winner. For example, suppose we have a list of k different voting rules. If the candidate names (viewed as natural numbers) are congruent, modulo k, to i then we use the ith voting rule. Such a form of hybridization gives elections which are often computationally difficult to control [27].

8 Manipulation on average

The fact that a particular voting rule is NP-hard to manipulate or to control only indicates that the problem is computationally intractable in the worst case. Assuming P≠NP, this means that there are pathological instances of the problem which will take exponential time to solve. It does not tell us whether the computational problem is hard on average or in practice. Unfortunately, several recent theoretical results suggest that elections are often easy to manipulate on average.

For example, Procaccia and Rosenschein proved that for most scoring rules and a wide variety of distributions over votes, when the size of the coalition is $o(\sqrt{n})$, the probability that they can change the result tends to 0, and when it is $\omega(\sqrt{n})$, the probability that they can manipulate the result tends to 1 [28]. They offer two interpretations of this result. On the positive side, they suggest it may focus attention on other distributions which are computationally hard to manipulate. On the negative side, they suggest that it may strengthen the argument that manipulation problems are easy on average.

As a second example, Xia and Conitzer have shown that for a large class of voting rules including STV, as the number of agents grows, either the probability that a coalition can manipulate the result is very small (as the coalition is too small), or the probability that they can easily manipulate the result to make any alternative win is very large [29]. They left open only a small interval in the size for the coalition for which the coalition is large enough to manipulate but not obviously large enough to manipulate the result easily. Such results raise concerns about whether computational complexity is really a barrier to manipulation in practice.

These concerns are strengthened by results about the probability of success of approximate or heuristic methods. Such methods construct simple greedy or random manipulations that can be shown to succeed with high probability in many circumstances. For example, Procaccia and Rosenschein give a simple greedy procedure that will in polynomial time find a manipulation of a scoring rule for any "junta" distribution of weighted votes with a probability of failure that is an inverse polynomial in n [30]. A "junta" distribution is concentrated on the hard instances.

As a second example, Friedgut et al. prove that if the voting rule is neutral and far from any dictatorship and there are three candidates then there exists an agent for whom a random manipulation succeeds with probability $\Omega(\frac{1}{n})$ where n is the number of agents [31]. They were unable to extend their proof to four (or more) candidates. On the other hand, Xia and Conitzer showed that, starting from different assumptions, a random manipulation would succeed with probability $\Omega(\frac{1}{n})$ for three or more candidates for STV, for four or more candidates for a scoring rule and for five or more candidates for Copeland [32].

9 Parameterized complexity

Another approach to gain insight into the computational complexity of manipulation comes from parameterized complexity. Such results can show that computational intractability depends on a particular parameter, like the number of candidates, being allowed to grow. Traditional worst case analysis focuses on those polynomial problems (for which the running time of the best algorithm is of the form $O(n^c)$ where n is the size of the input and c is some constant) and those problems which are NP-hard (for which the running time of the best known algorithms is not polynomial). This is essentially an one-dimensional view of complexity in terms of the single parameter, the problem size n. Parameterized complexity takes a more refined two-dimensional view of complexity. We try to identify another parameter of the problem such that the computational complexity is restricted to the size of this parameter. More precisely, we say that a problem is *fixed-parameter tractable* (FPT) if it can be solved in $O(f(k)n^c)$ time where f is *any* computable function, k is this new parameter, c is again a constant, and n remains the size of the input. If k is small and bounded in size, then the time to solve the problem remains essentially polynomial.

Not all problems are known to be fixed-parameter tractable. Downey and Fellows have proposed therefore a hierarchy of fixed-parameter *intractable* problem classes [33]:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq XP$$

For instance, the clique problem is $W[1]$-complete with respect to the size of the clique, whilst the dominating set problem is $W[2]$-complete with respect to the size of the dominating set. The class $W[t]$ is characterized by the depth t of unbounded fan-in gates in a Boolean circuit specifying the problem. There is considerable evidence to suggest that $W[1]$-hardness implies parametric intractability. In particular, the halting problem for non-deterministic Turing machines is $W[1]$-complete with respect to the length of the accepting computation.

Returning to the computational complexity of manipulation and control, both fixed-parameter tractability and intractability results are known. For instance, the control of Copeland elections by the addition or deletion of agents is fixed-parameter tractable when we bound either the number of candidates or the number of agents [34]. On the other hand, both destructive and constructive control of plurality voting by adding candidates are fixed-parameter intractable (specifically W[2]-hard) with respect to the number of added candidates [35]. Such results give additional insight into the source of computational complexity, and may help understand whether complexity is an issue in a particular case or not.

10 Manipulation in practice

Empirical studies have also considered the computational difficult of computing manipulations in practice. These studies reinforce the concerns raised by theoretical results. For example, Coleman and Teague experimentally demonstrated that only

relatively small coalitions are needed to change the elimination order of the STV rule in most cases [36]. On the other hand, they observed that most random elections are not trivially manipulable using a simple greedy heuristic. However, more complex methods appear able to compute manipulations in many cases relatively quickly. For instance, we showed that we compute manipulations of the STV rule for a wide range of random and non-random voting distributions in a reasonable amount of computation time even with hundreds of candidates using a simple recursive algorithm [37].

Most recently, we have conducted an extensive empirical investigation into the computational difficulty of manipulation [37–39]. Our experiments have covered two settings which represent the two types of complexity results that have been identified in this area: manipulation with unweighted votes by a single agent, and manipulation with weighted votes by a coalition of agents. In the first case, we considered the STV rule, and in the second case, the veto rule. STV is one of the few voting rules used in practice where it is NP-hard to compute how a single agent manipulates the result when votes are unweighted. It also appears one of the harder voting rules to manipulate since it involves multiple rounds and complex manipulating votes [40]. On the other hand, the veto rule is one of the simplest representatives of voting rules where it is NP-hard to compute how a coalition of weighted agents manipulates the result. The veto rule is also interesting from a complexity perspective as it is on the borderline of (in)tractability. Constructive manipulation of the veto rule by a coalition of weighted agents is NP-hard but destructive manipulation is polynomial [10].

Our experiments sampled a number of different distributions of votes including uniform, correlated and real world elections using the sampling techniques advocated in [41, 42]. The experiments look for so called "phase transition" behaviour. This has proven very successful for identifying hard instances of many other NP-complete problems including propositional satisfiability [43–46], constraint satisfaction [47–50], graph coloring [51–53], number partitioning [54–56] and the traveling salesperson problem [57, 58]. It has even proved useful in other complexity classes [59–61] as well as in optimisation and approximation problems [62–65]. On the other hand, care has to be taken not to generate "phase transition" instances that are flawed and thereby easy to solve [66–69].

In many of the elections in our experiments, it was easy to compute how to manipulate the result or to prove that manipulation was impossible. Whilst manipulations were often easy to compute, we did identify particular situations where it was computationally more difficult. For example, we observed difficult manipulation problems with veto voting when votes are highly correlated and the election was "hung". However, the addition of even a single uncorrelated agent was enough to make manipulation computationally easy again. Interestingly many of the probability curve we observed did not appear to sharpen to a step function around a fixed point. The probability curves often resembled the smooth phase transitions seen in polynomial problems like 2-coloring [70] and 1-in-2 satisfiability [61]. Our empirical results suggest that we need to treat computational complexity results about the NP-hardness of manipulation with some care. We were often able to compute a manipulation successfully using rather simple and direct methods without significant computational difficulty.

11 Single peaked preferences

Another escape from the Gibbard-Sattertwhaite theorem is to give up the notion of an universal domain and to limit the form of preferences that can be expressed by agents. For example, one such restriction is to *single peaked* preferences. Preferences are single peaked if we can put an order on the candidates, every agent has a preferred candidate and likes a candidate less as we move away from their preferred candidate. Single peaked preferences are natural when we are considering features like price (e.g. "I have a preferred budget to spend on an apartment, and the more I have to under-spend or over-spend, the more I will dislike the choice").

With single peaked preferences, the median voting rule (which elects the candidate for whom 50% of agents most prefer this candidate or a candidate to the left of this candidate) is strategy proof, surjective and not dictatorial. It is in the best interests of every agent to announce their most preferred candidate. If they vote for any other candidate, they either do not change the result or make it worse for themselves. Unfortunately, the voting rule may be fixed in advance and we may not be able to change it to the median rule when it becomes known that preferences are single peaked. In addition, the median voting rule is vulnerable to many other types of manipulation, and remains so when votes are single peaked. For example, it is computationally vulnerable to the addition of new agents. It may not therefore be appropriate or desirable to move to median voting. Hence, even when votes are single peaked, we may still have to worry about strategic voting.

Suppose that we have a voting rule like STV but preferences are single peaked. Does computational complexity remain a barrier to manipulation in this restricted setting? In [71], we proved that STV remains NP-hard to manipulate constructively or destructively with weighted votes and three or more candidates when votes are single peaked. Subsequently, Faliszewski et al. have studied the computational complexity of manipulating many other voting rules when preferences are single peaked [72]. They prove that many resistances to manipulation and control disappear when preferences are single peaked. For instance, it is NP-hard to compute a constructive manipulation of the veto rule when votes are weighted and there are three or more candidates. However, when preferences are single peaked, computing a manipulation is polynomial. They also show a surprising result where increasing the number of candidates decreases the computational complexity of computing a manipulation. More precisely, with single-peaked preferences, weighted votes and the 3-veto voting rule (i.e., each agent vetoes three candidates), computing a manipulation is polynomial for up to four candidates, NP-hard for five candidates, and polynomial again for six or more candidates. In general we conclude that structure in the preferences of agents can make manipulation easier to compute.

12 Mechanism design

Another direction in which to study manipulation comes from game theory. For example, *mechanism design* considers the design of systems in which agents are motivated to reveal private information like their preferences. A game is *incentive compatible* if all of the participants do best when they truthfully reveal any private

information asked for by the mechanism. Any manipulable voting system is not incentive compatible. On the other hand, if votes are single-peaked, then the median voting rule is incentive compatible. This suggests another escape from manipulation. The Gibbard–Sattertwhaite theorem tells us that truth-telling is not a dominant strategy. That is, it may not be the best strategy for one agent to reveal their true preferences irrespective of how the other agents play. We might instead look for a weaker condition. For example, are there situations where truth-telling is merely a Nash equilibrium? That is, are there situations where there is no incentive for an agent not to tell the truth about their preferences assuming the other agents also tell the truth?

13 Stable marriage problems

Another multi-agent problem in which manipulation may be an issue is the stable marriage problem. This is the well-known problem of matching men to women so that no man and woman who are not married to each other both prefer each other. It has a wide variety of practical applications such as a matching doctors to hospitals. As a practical example, the US Navy has a web-based multi-agent system for assigning sailors to ships [73]. As with voting, an important issue is whether agents can manipulate the result by mis-reporting their preferences. Unfortunately, Roth has proved that *all* stable marriage procedures can be manipulated [74].

We might hope that computational complexity might also be a barrier to manipulate stable marriage procedures. For example, we have proposed a new stable marriage procedures that is NP-hard to manipulate [75]. An advantage of our new procedure is that, unlike the Gale–Shapley algorithm, it is gender neutral since it does not favour one sex over the other. Our new procedure picks the stable matching that is most preferred by the most popular men and women. The most preferred men and women are chosen using a voting rule. We proved that, if the voting rule used is STV then the resulting stable matching procedure is also NP-hard to manipulate. We conjecture that other voting rules which are NP-hard to manipulate will give rise to stable matching procedures which are also NP-hard to manipulate. Of course, these are again only worst case results. It remains to be seen if such stable marriage problems are computationally difficult to manipulate on average or in practice.

14 Tournament manipulation

Another domain in which computational issues surrounding manipulation can be important is the domain of (sporting) tournaments. Manipulating a tournament is slightly different to manipulating an election. In a sporting tournament, the agents are also the candidates. Since it is hard (without bribery or similar mechanisms) for a team to play better than it can, one restriction is to consider just those manipulations where the manipulators can throw games. By comparison, in an election, agents in the manipulating coalition can mis-report their preferences in any way they choose. Unfortunately, we have shown that we can decide how to manipulate the two most popular types of sporting competitions (round robin and cup competitions) by throwing games in polynomial time [23]. In addition, we show that finding the

minimal number of games that need to be thrown to manipulate the result can also be determined in polynomial time. We also give a polynomial time proceure to calculate the probability that a team wins a cup competition under manipulation.

Another way in which a sporting tournament like a cup can be manipulated is by changing the agenda, the schedule of matches that are to be played. For example, Vu et al. prove that, given probabilities for the result of all the possible matches, it is NP-hard to compute an agenda for a balanced tournament so that a given agent wins with some probability [76]. A balanced tournament is one in which no agent plays more than one more match than any other agent. As a second example, Willians proves that, if only certain matches are allowed, then computing an agenda for a balanced tournament so that a particular agent wins is NP-hard [24]. Note that when the match results are deterministic and all matches are allowed, the computational complexity of the manipulation problem is an open problem.

15 Preference elicitation

A final domain that is closely connected to manipulation is preference elicitation. When we do not need to elicit any more preferences (because the winner cannot change), the remaining voters cannot manipulate the result. Eliciting preferences takes time and effort. In addition, voters may be reluctant to reveal all their preferences due to privacy and other concerns. We therefore may want to stop elicitation as soon as one candidate has enough support that they must win regardless of any missing preferences. We can consider the computational problem of deciding when we can stop eliciting preferences [71, 77, 78]. Like manipulation, this problem has been considered along a number of dimensions: weighted or unweighted votes, bounded or unbounded number of candidates, and coarse or fine elicitation (i.e., do we elicit a total ordering over the candidates from each voter, or a voter's preferences between two candidates?). For example, deciding if coarse elicitation is over is coNP-complete for STV with unweighted votes and an unbounded number of candidates but polynomial for plurality and Borda [77]. As a second example, for the cup rule on weighted votes, deciding if fine elicitation is over is coNP-complete when there are four or more candidates, whilst deciding if coarse elicitation is over is polynomial irrespective of the number of candidates [78]. This demonstrates that what we ask voters can be critical.

With partially elicited preferences, we can define the *possible winners* (those candidates who could still possibly win) as well as the *necessary winner* (the candidate who must now necessarily win) [79]. The necessary winner is always a member of the set of possible winners. When the possible winners equals the unit set containing the necessary winner, we can stop eliciting preferences. We can consider the computational problem of computing these sets [71]. Again, this has been considered along a number of dimensions like weighted or unweighted votes and a bounded or unbounded number of candidates. For example, with weighted votes, computing the set of possible winners for the Borda, Copeland or STV rules is NP-hard [3].

The problems of computing necessary and possible winners can be in different complexity classes. For instance, computing the set of possible winners for the second-order Copeland rule with unweighted votes and an unbounded number of candidates is NP-hard, whilst computing the necessary winner is polynomial. In

addition, even computing good approximations to the set of possible or necessary winners can be shown to be NP-hard [3]. However, in those cases like the Borda rule where computing the set of possible winners is polynomial, we can focus preference elicitation on those candidates that can still win [3]. This can potentially reduce the questions that we must ask of the voters.

Preference elicitation is one example of where we have uncertainty. In this case, the uncertainty is in the votes. There are, however, other types of uncertainty. The notions of possible and necessary winner are useful in other situations involving uncertainty. For example, when using the cup rule, there might be uncertainty about the agenda to be used [22]. Indeed, the chair might be keeping the agenda secret to make manipulation more difficult. Most recently, we have started to consider the complexity of computing manipulations when agents have partial information of a particular form about the votes [80].

16 Conclusions

We have surveyed recent results in the literature about whether computational complexity is a barrier to the manipulation of voting rules. NP-hardness results have been identified in two different settings: manipulation with unweighted votes by a single agent, and manipulation with weighted votes by a coalition of agents. In addition, tie-breaking and techniques like hybridizing together two or more voting rules have been shown to add to the computational difficulty of computing a manipulation. Unfortunately, NP-hardness results only bound the worst case. We have argued that a number of average case and empirical results suggest that manipulations may be easy to compute in practice. Simple greedy and random procedures are often able to compute manipulations with a high probability of success. In addition, structure in the preferences of agents like single peakedness also often reduces the computational complexity of computing manipulations. Our overall conclusion is that computational complexity appears to be a rather weak and limited barrier to manipulation.

Are there any places that computational complexity might still lie waiting to make manipulation difficult to compute? One area that has received only a limited amount of attention is when we have uncertainty in the votes. Many of our results have assumed complete knowledge of how the non-manipulating agents will vote. If we have only probabilistic information, it may be difficult to compute how to manipulate the result as we need essentially to consider many different scenarios. Whilst there are some theoretical worst-case results about computing manipulations in the presence of uncertainty, there is limited evidence yet about the computational difficulty in practice. Finally, it would be interesting to consider related problems like preference elicitation where computational complexity has also been identified but where, unlike manipulation, computational complexity is typically more unwelcome [71, 78].

Acknowledgements NICTA is funded by the Australian Government through the Department of Broadband, Communications and the Digital Economy, and the Australian Research Council. The author is also supported by the Asian Office of Aerospace Research and Development (AOARD-104123).

References

1. Lowrie, P.R.: The Sydney coordinated adaptive traffic system : principles, methodology, algorithms. In: Proceedings of International Conference on Road Traffic Signaling, pp. 67–70. Institution of Electrical Engineers, London, U.K. (1982)
2. Rossi, F., Venable, B., Walsh, T.: mCP nets: representing and reasoning with preferences of multiple agents. In: Proceedings of the 19th National Conference on AI. Association for Advancement of Artificial Intelligence (2004)
3. Pini, M., Rossi, F., Venable, B., Walsh, T.: Incompleteness and incomparability in preference aggregation. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007). International Joint Conference on Artificial Intelligence (2007)
4. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Llull and Copeland voting broadly resist bribery and control. In: Proceedings of the 22nd National Conference on AI, pp. 724–730. Association for Advancement of Artificial Intelligence (2007)
5. Moulin, H.: Axioms of cooperative decision making. Cambridge University Press, United Kingdom (1988)
6. Gibbard, A.: Manipulation of voting schemes: a general result. Econometrica **41**, 587–601 (1973)
7. Satterthwaite, M.: Strategy-proofness and arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. J. Econ. Theory **10**, 187–216 (1975)
8. Bartholdi, J.J., Tovey, C.A., Trick, M.A.: The computational difficulty of manipulating an election. Soc. Choice Welf. **6**(3), 227–241 (1989)
9. Bartholdi, J.J., Orlin, J.B.: Single transferable vote resists strategic voting. Soc. Choice Welf. **8**(4), 341–354 (1991)
10. Conitzer, V., Sandholm, T., Lang, J.: When are elections with few candidates hard to manipulate? J. Assoc. Comput. Mach. **54**(3) (2007) (Article 14 (33 pages))
11. Conitzer, V., Sandholm, T.: Complexity of manipulating elections with few candidates. In: Dechter, R., Kearns, M., Sutton, R. (eds.) Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002), pp. 314–319. Association for Advancement of Artificial Intelligence (2002)
12. Faliszewski, P., Hemaspaandra, E., Schnoor, H.: Copeland voting: ties matter. In: Padgham, L., Parkes, D.C., Müller, J.M., Parsons, S. (eds.) 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 983–990 (2008)
13. Davies, J., Katsirelos, G., Narodytska, N., Walsh, T.: Complexity of and Algorithms for Borda Manipulation. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011). AAAI Press (2011)
14. Betzler, N., Niedermeier, R., Woeginger, G.J.: Unweighted coalitional manipulation under the Borda rule is NP-hard. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011) (2011)
15. Narodytska, N., Walsh, T., Xia, L.: Manipulation of Nanson's and Baldwin's rules. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011). AAAI Press (2011)
16. Brelsford, E., Faliszewski, P., Hemaspaandra, E., Schnoor, H., Schnoor, I.: Approximability of manipulating elections. In: Fox, D., Gomes, C.P. (eds.) Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008), pp. 44–49. AAAI Press (2008)
17. Zuckerman, M., Procaccia, A.D., Rosenschein, J.S.: Algorithms for the coalitional manipulation problem. In: Teng, S.H. (ed.) Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 277–286 (2008)
18. Davies, J., Katsirelos, G., Narodytska, N., Walsh, T.: An empirical study of Borda manipulation. In: Conitzer, V., Rother, J. (eds.) Proc. of the Third International Workshop on Computational Social Choice, pp. 91–102. Düsseldorf University Press (2010)
19. Bartholdi, J.J., Tovey, C.A., Trick, M.A.: How hard is it to control an election. Math. Comput. Model. **16**(8–9), 27–40 (1992)
20. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A.: Multimode control attacks on elections. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009), pp. 128–133 (2009)
21. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A.: How hard is bribery in elections? J. Artif. Intell. Res. (JAIR) **35**, 485–532 (2009)
22. Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Dealing with incomplete agents' preferences and an uncertain agenda in group decision making via sequential majority voting. In: Brewka,

G., Lang, J. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference (KR 2008), pp. 571–578. AAAI Press (2008)
23. Russell, T., Walsh, T.: Manipulating tournaments in cup and round robin competitions. In: Rossi, F., Tsoukiàs, A. (eds.) Algorithmic Decision Theory, First International Conference (ADT 2009). Lecture Notes in Computer Science, vol. 5783, pp. 26–37. Springer (2009)
24. Williams, V.V.: Fixing a Tournament. In: Fox, M., Poole, D. (eds.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010). AAAI Press (2010)
25. Conitzer, V., Sandholm, T.: Universal voting protocol tweaks to make manipulation hard. In: Proceedings of 18th IJCAI. International Joint Conference on Artificial Intelligence, pp. 781–788 (2003)
26. Elkind, E., Lipmaa, H.: Hybrid voting protocols and hardness of manipulation. In: Deng, X., Du, D.-Z. (eds.) Proceedings of 16th International Symposium on Algorithms and Computation (ISAAC 2005). Lecture Notes in Computer Science, vol. 3827, pp. 206–215. Springer (2005)
27. Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Hybrid elections broaden complexity-theoretic resistance to control. Math. Log. Q. **55**(4), 397–424 (2009)
28. Procaccia, A.D., Rosenschein, J.S.: Average-case tractability of manipulation in voting via the fraction of manipulators. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) Proceedings of 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-07), pp. 718–720. IFAAMAS (2007)
29. Xia, L., Conitzer, V.: Generalized scoring rules and the frequency of coalitional manipulability. In: Fortnow, L., Riedl, J., Sandholm, T. (eds.) EC '08: Proceedings of the 9th ACM Conference on Electronic Commerce, pp. 109–118. ACM (2008)
30. Procaccia, A.D., Rosenschein, J.S.: Junta distributions and the average-case complexity of manipulating elections. J. Artif. Intell. Res. **28**, 157–181 (2007)
31. Friedgut, E., Kalai, G., Nisan, N.: Elections can be manipulated often. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 243–249. IEEE Computer Society Press (2008)
32. Xia, L., Conitzer, V.: A sufficient condition for voting rules to be frequently manipulable. In: Fortnow, L., Riedl, J., Sandholm, T. (eds.) Proceedings of the 9th ACM conference on Electronic Commerce (EC' 08), pp. 99–108. ACM (2008)
33. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized complexity: a framework for systematically confronting computational intractability. In: Graham, R., Kratochvil, J., Nesetril, J., Roberts, F. (eds.) Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 49, pp. 49–99. American Mathematical Society (1999)
34. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: Llull and Copeland voting computationally resist bribery and constructive control. J. Artif. Intell. Res. (JAIR) **35**, 275–341 (2009)
35. Betzler, N., Uhlmann, J.: Parameterized complexity of candidate control in elections and related digraph problems. Theor. Comp. Sci. **410**(52), 5425–5442 (2009)
36. Coleman, T., Teague, V.: On the complexity of manipulating elections. In: Gudmundsson, J., Jay, B. (eds.) Proceedings of the 13th Australasian Symposium on Theory of Computing (CATS '07), pp. 25–33. Australian Computer Society, Inc. (2007)
37. Walsh, T.: An empirical study of the manipulability of single transferable voting. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Proc. of the 19th European Conference on Artificial Intelligence (ECAI-2010). Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 257–262. IOS Press (2010)
38. Walsh, T.: Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009), pp. 324–329 (2009)
39. Walsh, T.: Where are the hard manipulation problems? J. Artif. Intell. Res. (2011, to appear)
40. Chamberlin, J.: An investigation into the relative manipulability of four voting systems. Behav. Sci. **30**, 195–203 (1985)
41. Gent, I.P., Walsh, T.: Phase transitions from real computational problems. In: Proceedings of the 8th International Symposium on Artificial Intelligence: Intelligent Systems Applications in Industry and Business, pp. 356–364 (1995)
42. Gent, I.P., Hoos, H., Prosser, P., Walsh, T.: Morphing: combining structure and randomness. In: Hendler, J., Subramanian, D. (eds.) Proceedings of the 16th National Conference on Artificial Intelligence (AAAI 1999), pp. 654–660. Association for Advancement of Artificial Intelligence (1999)

43. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. In: Proceedings of the 10th National Conference on AI, pp. 459–465. Association for Advancement of Artificial Intelligence (1992)
44. Gent, I.P., Walsh, T.: The SAT phase transition. In: Cohn, A.G. (ed.) Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94), pp. 105–109. Wiley (1994)
45. Gent, I., Walsh, T.: Easy problems are sometimes hard. Artif. Intell. **70**, 335–345 (1994)
46. Gent, I.P., Walsh, T.: The satisfiability constraint gap. Artif. Intell. **81**(1–2), 59–80 (1996)
47. Prosser, P.: Binary constraint satisfaction problems: some are harder than others. In: Cohn, A.G. (ed.) Proceedings of the 11th European Conference on Artificial Intelligence. European Conference on Artificial Intelligence, pp. 95–99. Wiley (1994)
48. Smith, B.: The phase transition in constraint satisfaction problems: a closer look at the mushy region. In: Cohn, A.G. (ed.) Proceedings of the 11th European Conference on Artificial Intelligence. European Conference on Artificial Intelligence, pp. 100–104. Wiley (1994)
49. Gent, I.P., MacIntyre, E., Prosser, P., Walsh, T.: Scaling effects in the CSP phase transition. In: Montanari, U., Rossi, F. (eds.) Proceedings of the 1st International Conference on Principles and Practices of Constraint Programming (CP-95). Lecture Notes in Computer Science, vol. 976, pp. 70–87. Springer (1995)
50. Gent, I.P., MacIntyre, E., Prosser, P., Walsh, T.: The constrainedness of search. In: Proceedings of the 13th National Conference on AI. Association for Advancement of Artificial Intelligence, pp. 246–252 (1996)
51. Walsh, T.: The constrainedness knife-edge. In: Mostow, J., Rich, C. (eds.) Proceedings of the 15th National Conference on AI. Association for Advancement of Artificial Intelligence, pp. 406–411 (1998)
52. Walsh, T.: Search in a small world. In: Dean, T. (ed.) Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99), pp. 1172–1177. Morgan Kaufmann (1999)
53. Walsh, T.: Search on high degree graphs. In: Nebel, B. (ed.) Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), pp. 266–274. Morgan Kaufmann (2001)
54. Gent, I.P., Walsh, T.: Phase transitions and annealed theories: number partitioning as a case study. In: Wahlster, W. (ed.) Proc. of the 12th European Conference on Artificial Intelligence (ECAI-96). Wiley, Chichester, pp. 170–174 (1996)
55. Gent, I.P., Walsh, T.: Analysis of heuristics for number partitioning. Comput. Intell. **14**(3), 430–451 (1998)
56. Mertens, S.: A physicist's approach to number partitioning. Theor. Comp. Sci. **265**(1–2), 79–108 (2001)
57. Zhang, W., Korf, R.E.: A study of complexity transitions on the asymmetic traveling salesman problem. Artif. Intell. **81**(1–2), 223–239 (1996)
58. Gent, I.P., Walsh, T.: The TSP phase transition. Artif. Intell. **88**, 349–358 (1996)
59. Gent, I.P., Walsh, T.: Beyond NP: the QSAT phase transition. In: Hendler, J., Subramanian, D. (eds.) Proceedings of the 16th National Conference on AI. Association for Advancement of Artificial Intelligence, pp. 648–653 (1999)
60. Gent, I.P., MacIntyre, E., Prosser, P., Shaw, P., Walsh, T.: The constrainedness of arc consistency. In: 3rd International Conference on Principles and Practices of Constraint Programming (CP-97), pp. 327–340. Springer (1997)
61. Walsh, T.: From P to NP: COL, XOR, NAE, 1-in-k, and Horn SAT. In: Dechter, R., Kearns, M., Sutton, R. (eds.) Proceedings of the 17th National Conference on AI (AAAI 2002). Association for Advancement of Artificial Intelligence, pp. 695–700 (2002)
62. Slaney, J., Walsh, T.: Backbones in optimization and approximation. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001) (2001)
63. Slaney, J., Walsh, T.: Phase transition behavior: from decision to optimization. In: Proceedings of the 5th International Symposium on the Theory and Applications of Satisfiability Testing, SAT (2002)
64. Kilby, P., Slaney, J., Thiebaux, S., Walsh, T.: Backbones and backdoors in satisfiability. In: Veloso, M.M., Kambhampati S. (eds.) Proceedings of the 20th National Conference on AI. Association for Advancement of Artificial Intelligence, pp. 1368–1373 (2005)
65. Kilby, P., Slaney, J., Walsh, T.: The Backbone of the travelling salesperson. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005), pp. 175–180 (2005)
66. Gent, I.P., Grant, S.A., MacIntyre, E., Prosser, P., Shaw, P., Smith, B.M., Walsh, T.: How not to do it. Research Report 97.27, School of Computer Studies, University of Leeds (1997) (An earlier and shorter version of this report by the first and last authors appears in Proceedings of

the AAAI-94 Workshop on Experimental Evaluation of Reasoning and Search Methods and as Research Paper No 714, Dept. of Artificial Intelligence, Edinburgh (1994))
67. MacIntyre, E., Prosser, P., Smith, B., Walsh, T.: Random constraint satisfaction: theory meets practice. In: 4th International Conference on Principles and Practices of Constraint Programming (CP-98), pp. 325–339. Springer (1998)
68. Gent, I.P., MacIntyre, E., Prosser, P., Smith, B.M., Walsh, T.: Random constraint satisfaction: flaws and structure. Constraints **6**(4), 345–372 (2001)
69. Gomes, G., Walsh, T.: Randomness and structure. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Foundations of Artificial Intelligence, pp. 639–664. Elsevier (2006)
70. Achlioptas, D.: Threshold phenomena in random graph colouring and satisfiability. Ph.D. thesis, Department of Computer Science, University of Toronto (1999)
71. Walsh, T.: Uncertainty in preference elicitation and aggregation. In: Proceedings of the 22nd National Conference on AI. Association for Advancement of Artificial Intelligence, pp. 3–8 (2007)
72. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.A., Rothe, J.: The shield that never was: societies with single-peaked preferences are more open to manipulation and control. In: Heifetz, A. (ed.) Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2009), pp. 118–127 (2009)
73. Liebowitz, J., Simien, J.: Computational efficiencies for multi-agents: a look at a multi-agent system for sailor assignment. Electronc Government **2**(4), 384–402 (2005)
74. Roth, A.E.: The economics of matching: stability and incentives. Math. Oper. Res. **7**, 617–628 (1982)
75. Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Manipulation and gender neutrality in stable marriage procedures. In: 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009). pp. 665–672 (2009)
76. Vu, T., Altman, A., Shoham, Y.: On the complexity of schedule control problems for knockout tournaments. In: Sierra, C., Castelfranchi, C., Decker, K.S., Sichman, J.S. (eds.) 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 225–232. IFAAMAS (2009)
77. Conitzer, V., Sandholm, T.: Vote elicitation: complexity and strategy-proofness. In: Dechter, R., Kearns, M., Sutton. R. (eds.) Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002). Association for Advancement of Artificial Intelligence, pp. 392–397 (2002)
78. Walsh, T.: Complexity of terminating preference elicitation. In: Padgham, L., Parkes, D.C., Müller, J.P., Parsons S. (eds.) 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 967–974. IFAAMAS (2008)
79. Konczak, K., Lang, J.: Voting procedures with incomplete preferences. In: Proceedings of the IJCAI-2005 workshop on Advances in Preference Handling (2005)
80. Conitzer, V., Walsh, T., Xia, L.: Dominating manipulations in voting wih partial information. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011). AAAI Press (2011)