A Scheduling Approach to Coalitional Manipulation

Lirong Xia
Department of Computer Science
Duke University
Durham, NC 27708, USA
lxia@cs.duke.edu

Vincent Conitzer
Department of Computer Science
Duke University
Durham, NC 27708, USA
conitzer@cs.duke.edu

Ariel D. Procaccia
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA
arielpro@seas.harvard.edu

ABSTRACT

The coalitional manipulation problem is one of the central problems in computational social choice. In this paper, we focus on solving the problem under the important family of positional scoring rules, in an approximate sense that was advocated by Zuckerman et al. [SODA 2008, AIJ 2009]. Our main result is a polynomial-time algorithm with (roughly speaking) the following theoretical guarantee: given a manipulable instance with m alternatives, the algorithm finds a successful manipulation with at most $m-2$ additional manipulators. Our technique is based on a reduction to the scheduling problem known as $Q|pmtn|C_{max}$, along with a novel rounding procedure. We demonstrate that our analysis is tight by establishing a new type of integrality gap. We also resolve a known open question in computational social choice by showing that the coalitional manipulation problem remains (strongly) NP-complete for positional scoring rules even when votes are unweighted. Finally, we discuss the implications of our results with respect to the question: "Is there a prominent voting rule that is usually hard to manipulate?"

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Economics, Theory

Keywords

Social choice, Coalitional manipulation, Positional scoring rules, Scheduling

1. INTRODUCTION

In settings with multiple agents, the agents often need to make a group decision regarding a set of alternatives (also called "candidates"). A natural way of doing this is by *voting*. Each agent (also known as *voter*) is asked to submit a linear order over the set of alternatives, which represents her preferences, and a winning alternative is determined by applying a *voting rule* to the collection of submitted linear orders.

Ideally the voters would submit linear orders that represent their true preferences. However, sometimes a voter can submit a false vote (an order that does not represent her true preferences) that makes her better off. This phenomenon is called *manipulation*, and the culprit is called a *manipulator*. If, under voting rule r, there is no instance where a voter benefits from manipulation, then r is said to be *strategy-proof*.

Unfortunately, it is impossible to design a strategy-proof voting rule that satisfies some very basic additional properties, due to the well-known Gibbard-Satterthwaite theorem [12, 18] (see [15] for an overview). To get around this very negative result, it has been suggested to consider computational complexity as a barrier against manipulation. The idea is that the mere existence of an effective manipulation does not guarantee that the manipulators can find it in a reasonable amount of time. The computational complexity of manipulation in voting systems is one of the main research topics in the burgeoning field of *computational social choice*; numerous papers have been devoted to this problem, covering different rules and different assumptions regarding the manipulation setting.

In the earliest work on the complexity of manipulation [2, 1] it was shown that (if the number of alternatives is unbounded) it is NP-complete to determine whether a single manipulator can effectively manipulate the election, under both the second-order Copeland and the STV rules. Later work studied how to modify prominent voting rules in a way that makes them hard to manipulate for a single voter [4, 8].

A more general manipulation setting is that of *weighted coalitional manipulation (WCM)*. In this setting, multiple manipulators have formed a coalition, with the goal of making an agreed-upon alternative win the election. Furthermore, the voters in this setting are weighted, where a voter with weight k is equivalent to k unweighted voters that cast identical ballots. Weights are common, e.g., in corporate elections, where voters are weighted according to the amount of stock they hold. Conitzer et al. [6] have established that this problem is computationally hard under a variety of prominent voting rules, even when the number of alternatives is constant.

Subsequent work by Hemaspaandra et al. [14] has dealt with *positional scoring rules*. Each rule in this family can be represented by a vector $\vec{s} = (s_1, \ldots, s_m)$. Each voter then awards s_i points to the alternative that she ranks in the ith position; the alternative with most points wins the election. This family includes three of the most prominent voting rules: *Plurality* (where each voter awards one point to her favorite alternative), *Borda* (where each voter awards $m-i$ points to the alternative ranked ith), and *Veto*

EC'10, June 7–11, 2010, Cambridge, Massachusetts, USA.

(where each voter awards one point to all the alternatives, except for the last-ranked one). Hemaspaandra et al. have established a dichotomy theorem for WCM in scoring rules. This theorem classifies whether WCM is NP-complete or in P, depending on the parameters $\vec{s}$.

A special case of weighted coalitional manipulation is its *unweighted* version (UCM), which is perhaps more natural in most settings (e.g., political elections). Progress on the UCM problem has been significantly slower than on other variations, but many of the questions have recently been resolved. The exact complexity of the problem is now known with respect to almost all of the prominent voting rules [9, 22, 23], with the glaring exception of Borda. Researchers have believed for some years that UCM under Borda is NP-complete; this belief was explicitly put forward as a conjecture by Zuckerman et al. [23], but the question still remains open. In fact, although UCM under some positional scoring rules is known to be tractable (e.g., Plurality and Veto), previous work has failed to find any positional scoring rule for which UCM is hard.

The main thrust of the results of Zuckerman et al. [23] is the design of algorithms for WCM and UCM with unusual approximation guarantees. The most interesting of these results deals with Borda in the context of WCM: it is shown that a greedy manipulation algorithm has the curious property that given an instance of WCM under Borda where there is a manipulation, the algorithm is guaranteed to find a manipulation that requires only one additional manipulator with maximum weight, that is, with weight as large as any of the original manipulators. Furthermore, it is observed that the unweighted coalitional manipulation setting begs the consideration of a natural optimization problem, *unweighted coalitional optimization (UCO)*. In this problem, we are given the votes of an unweighted set of voters, and the goal is to determine the minimum number of manipulators needed to make a given alternative win the election. It follows from the result mentioned above that the greedy algorithm approximates UCO in Borda to an additive term of one.

Our results and techniques. In this paper, we focus on WCM, UCM, UCO, and COd (the problem that asks for the minimum weight that a single manipulator who can cast a *divisible* vote needs in order to make a given alternative win) under positional scoring rules; we look for approximability in the sense of Zuckerman et al. [23], as discussed above. Our main contribution is the exploration of a surprising and fruitful connection between coalitional manipulation and scheduling. We demonstrate that the huge body of work on the latter problem can be leveraged to obtain nontrivial algorithmic results for the former problem.

The intuition behind the reduction is as follows. The scheduling problem to which we reduce is that of scheduling on parallel machines where the goal is to minimize makespan. In the coalitional manipulation problem, each manipulator j always ranks the coalition's preferred alternative c first, but must award $s_i \cdot w_j$ points to the alternative it ranks ith, where w_j is the manipulator's weight. For any $i \geq 2$, we define a machine for s_i; the larger s_i is in relation to s_1, the slower the machine is. Furthermore, each alternative besides c is a job; the larger the gap between the score of this alternative and the score of c, the larger the job is. When a manipulator with weight w_j ranks an alternative in the ith position, it decreases the gap between c and this alternative by $(s_1 - s_i)w_j$ points, which, under the detailed reduction, is equivalent to processing the corresponding job on the $(i-1)$th slowest machine for w_j time units.

In Section 3.1, we consider a version of WCM where votes are divisible, that is, each voter is allowed to submit a convex combination of linear orders (instead of a single linear order, as in the traditional setting). This "divisible" variant of the problem is denoted by WCMd (whereas the traditional coalitional manipulation problem (the indivisible version) is denoted by WCM). WCMd may be interesting in its own right, but mainly serves to prepare the ground for our results regarding WCM. We give a polynomial-time algorithm for WCMd under any positional scoring rule by reducing it to the well-studied scheduling problem known as $Q|pmtn|C_{max}$ (in which preemptions are allowed). This algorithm also solves COd.

In Section 3.2 we deal with the indivisible case (WCM), and augment the WCMd algorithm with a rounding technique. Based on existing results from the scheduling literature, we can assume that the scheduling solutions use relatively few preemptive break points. We then show that in the coalitional manipulation problem, we need at most one additional voter per preemptive break point. We obtain the following theorem, which is a somewhat weaker but far more generally applicable version of the main result of Zuckerman et al. regarding Borda [23, Theorem 3.4].

THEOREM 3.9. *Algorithm 2 runs in polynomial time and*

1. *if the algorithm returns* false*, then there is no successful manipulation (even for the WCMd version of the instance);*
2. *otherwise, the algorithm returns a successful manipulation for a modified set of manipulators, consisting of the original manipulators plus at most $m-2$ additional manipulators, each with weight at most $W/2$, where W is the maximum weight of the manipulators.*

Crucially, in most settings of interest (e.g., political elections), the number of alternatives m is small compared to the number of voters, or even the number of manipulators. Moreover, WCM is NP-complete under scoring rules such as Borda and Veto, even when there are only three alternatives [6]. Therefore, in many important scenarios, $m-2$ additional manipulators constitute a very small fraction of the total number of manipulators, that is, the algorithm gives a good "approximation" to WCM.

A direct implication of Theorem 3.9 is that in the unweighted case (UCM) our approximation algorithm always finds a manipulation with at most $m-2$ additional manipulators, if there exists one for the given instance. Put another way, the algorithm approximates UCO to an additive term of $m-2$.

In Section 4, we establish an "integrality gap," in the following sense: the optimal solution to UCO can require $m-2$ more manipulators than the optimal solution to UCOd (Theorem 4.3). Moreover, we show that there is a family of instances of UCO such that any algorithm that is based on rounding an optimal solution for COd requires $m-2$ more votes than the optimal UCO solution (Theorem 4.4). These results suggest that the analysis of the guarantees provided by our technique is tight.

Our final major result asserts that UCM under a specific positional scoring rule is strongly NP-complete. (We note that in this context, this is actually a positive result, because manipulation is undesirable—just as the results earlier in the paper are actually negative results.) This shows that the answer to the previously open question of whether there exists an efficient algorithm that solves UCM under any positional scoring rule is "no" (assuming P$\neq$NP). While the problem remains open for Borda, the positional scoring rule for which we show the hardness result displays significant similarities to Borda; hence, we believe that our result gives strong support to the conjecture that UCM under Borda is hard. This result also justifies our approximation results for UCM and UCO, since it implies that we are approximating a problem that is indeed NP-hard. Moreover, our result implies that $Q|pmtn|C_{max}$ is strongly NP-complete in discrete time, that is, when preemptions are only allowed at integral times.[1]

[1]Pinedo [16], after asserting that the special case of this problem

Implications with respect to frequency of manipulation. Despite the large volume of work on worst-case hardness of manipulation, it is becoming increasingly clear that the question should be: "Is there a prominent voting rule that is *usually* hard to manipulate?" A stream of papers in recent years lends support to the belief that the answer is negative. It is possible to identify two main approaches.

One approach tries to define general basic properties that cannot be satisfied by any voting rule that is usually hard to manipulate (see, e.g., [10, 20, 7]). In this line of work, "usually" refers to using the uniform distribution for the preferences of the agents, known in the social choice literature as the *impartial culture assumption*. However, in realistic settings one would expect to encounter biased distributions that, e.g., favor specific alternatives or exhibit a concentration of voters around opposing camps.

A second approach strives to design efficient heuristic manipulation algorithms for prominent voting rules that are NP-hard to manipulate. Early work on this includes the work of Procaccia and Rosenschein [17] and Conitzer and Sandholm [5]. The work of Zuckerman et al. [23] also falls into this framework, but has the important advantage of allowing for theoretical guarantees without making any distributional assumptions.

Let us reconsider our Theorem 3.9 according to the perspective offered by Zuckerman et al. This theorem naturally applies to UCM (Corollary 4.1). Our algorithm might fail to correctly decide a given "no" instance of UCM (in the sense that it actually returns a solution, with additional manipulators), but given a similar instance with slightly fewer manipulators, the algorithm would answer "no". Hence, if m is relatively small, the algorithm's "window of error"—the family of instances on which the algorithm might fail—is relatively small. Without claiming anything formally, it would appear to follow that under "reasonable" distributions over preferences, the probability of drawing an instance on which the algorithm fails is small. Put another way, our results suggest that, although coalitional manipulation is NP-hard under prominent positional scoring rules, the problem is in fact usually quite easy under any positional scoring rule.

Future research. Several intriguing questions remain open. The first is whether a better additive approximation is possible.

OPEN QUESTION 1. *Is there a polynomial-time algorithm that gives an additive approximation of less than $m-2$ to UCO under all positional scoring rules?*

Our Theorem 4.4 shows that any such algorithm will have to use a fundamentally different technique. Another open problem is to understand the guarantees that the Greedy algorithm—the algorithm that was used by Zuckerman et al. [23] to prove the earlier result for Borda—gives with respect to general scoring rules.

OPEN QUESTION 2. *What additive approximation to UCO does the Greedy algorithm give for positional scoring rules?*

We also note that our Theorem 5.1 still has not resolved the complexity of UCM under Borda.

OPEN QUESTION 3. *Is UCM under Borda NP-complete?*

Finally, a more open-ended direction for future research is to investigate whether the techniques in this paper can be used for more general classes of voting rules (for example, generalized scoring rules [19, 21]). If so, this would cast further doubt on our ability to find a voting rule that is "usually" hard to manipulate.

where all machines have the same speed can be solved in polynomial time, claims that the results of his chapter can be extended to the case where they do not have the same speed (his Theorem 5.2.12). In light of our results, and based on a correspondence with the author, we conclude that this result is incorrect.

2. PRELIMINARIES

Let $\mathcal{C}$ be the set of *alternatives*. A linear order on $\mathcal{C}$ is a transitive, antisymmetric, and total relation on $\mathcal{C}$. The set of all linear orders on $\mathcal{C}$ is denoted by $L(\mathcal{C})$. The set of all convex combinations over $L(\mathcal{C})$ is denoted by $\Delta(L(\mathcal{C}))$. An *(indivisible) vote* is a linear order over $\mathcal{C}$, that is, it is an element of $L(\mathcal{C})$. A *divisible vote* is an element of $\Delta(L(\mathcal{C}))$. An n-voter indivisible profile P on $\mathcal{C}$ consists of n linear orders on $\mathcal{C}$, that is, $P = (R_1, \ldots, R_n)$, where for every $i \leq n$, $R_i \in L(\mathcal{C})$. Similarly, an n-voter divisible profile P on $\mathcal{C}$ consists of n convex combinations over $L(\mathcal{C})$. In the remainder of the paper we let m denote the number of alternatives (that is, $m = |\mathcal{C}|$), and let $\mathcal{C} = \{c, c_1, \ldots, c_{m-1}\}$.

A *voting rule* r is a function from the set of all indivisible profiles on $\mathcal{C}$ to nonempty subsets of $\mathcal{C}$, that is, the rule designates a nonempty subset of winners. A *(positional) scoring rule* over $\mathcal{C}$ is defined by a *scoring vector* $\vec{s} = (s_1, \ldots, s_m)$. For any linear order $V \in L(\mathcal{C})$ and any $c' \in \mathcal{C}$, let $s(V, c') = s_j$, where j is the rank of c' in V. For any $k \in \mathbb{N}$, any $V_1, \ldots, V_k \in L(\mathcal{C})$, and any $\alpha_1, \ldots, \alpha_k \geq 0$ such that $\sum_{i=1}^{k} \alpha_i = 1$, we let

$$s(\sum_{i=1}^{k} \alpha_k V_k, c') = \sum_{i=1}^{k} \alpha_k \cdot s(V_k, c').$$

For any profile $P = (V_1, \ldots, V_n)$, let $s(P, c') = \sum_{i=1}^{n} s(V_i, c')$. The rule selects alternatives $c' \in \mathcal{C}$ that maximize $s(P, c')$. Three prominent examples of scoring rules are *Borda*, for which the scoring vector is $(m-1, m-2, \ldots, 1, 0)$; *Plurality*, for which the scoring vector is $(1, 0, \ldots, 0, 0)$; and *Veto*, for which the scoring vector is $(1, 1, \ldots, 1, 0)$.

The definitions naturally extend to the case in which voters are weighted; the weights are represented by a vector $\vec{w} = (w_1, \ldots, w_n) \in \mathbb{R}_+^n$, where for any $i \leq n$, w_i is the weight of voter i. In particular, we let

$$s(P, \vec{w}, c') = \sum_{i=1}^{n} w_i \cdot s(V_i, c'),$$

and let $r(P, \vec{w})$ denote the set of winners (the alternatives with the highest score).

Let us now turn to the definition of the computational problems that we shall investigate. We study the so-called *constructive* manipulation variants, in which the goal is to make a given alternative win.[2]

DEFINITION 2.1. *The* Unweighted Coalitional Manipulation (UCM) *problem is defined as follows. An instance is a tuple (r, P^{NM}, c, k), where r is a voting rule, P^{NM} is the non-manipulators' profile, c is the alternative preferred by the manipulators, and k is the number of manipulators. We are asked whether there exists a profile P^M of indivisible votes for the manipulators such that $c \in r(P^{NM} \cup P^M)$.*

DEFINITION 2.2. *The* Weighted Coalitional Manipulation (WCM) *problem is defined as follows. An instance is a tuple*

[2]Contrast with the *destructive* versions of these problems, where the goal is to ensure that a given alternative does *not* win. The constructive versions are by far the more commonly studied ones, in part because an algorithm for a constructive version can be used to obtain an algorithm for a destructive version, simply by solving the constructive version for each other alternative.

$(r, P^{NM}, \vec{w}^{NM}, c, k, \vec{w}^{M})$, where r is a voting rule, P^{NM} is the non-manipulators' profile, $\vec{w}^{NM}$ represents the weights of P^{NM}, c is the alternative preferred by the manipulators, k is the number of manipulators, and $\vec{w}^{M} = (w_1, \ldots, w_k)$ represents the weights of the manipulators. We are asked whether there exists a profile P^{M} of indivisible votes for the manipulators such that $c \in r((P^{NM}, P^{M}), (\vec{w}^{NM}, \vec{w}^{M}))$.

Since we only focus on positional scoring rules in this paper, r will simply be represented by the scoring vector $(s_1, s_2, \ldots, s_m)$. In the above definitions, we use the *co-winner* formulation. Another possibility is to consider the *unique winner* formulation which is similar, only we require that the winning set be the singleton $\{c\}$, that is, $r((P^{NM}, P^{M}), (\vec{w}^{NM}, \vec{w}^{M})) = \{c\}$. Unless explicitly mentioned otherwise, our results hold for the unique winner formulation as well.

Zuckerman et al. [23] noted that the unweighted manipulation setting allows for a natural optimization problem: the *unweighted coalitional optimization* problem. Given, essentially, an unweighted coalitional manipulation instance, we ask *how many* manipulators are needed in order to make c win. Formally:

DEFINITION 2.3. The *Unweighted Coalitional Optimization (UCO) problem is defined as follows. An instance is a tuple (r, P^{NM}, c), where r is a voting rule, P^{NM} is the non-manipulators' profile, and c is the alternative preferred by the manipulators. We must find the minimum k such that there exists a a profile P^{M} consisting of k indivisible manipulator votes that satisfies $c \in r(P^{NM} \cup P^{M})$.*

In the weighted version of the optimization problem, we look for the minimum total *weight* of the manipulators that is sufficient to make c a co-winner.[3]

DEFINITION 2.4. *The* Coalitional Optimization for divisible votes (COd) *problem is defined as follows. An instance is a tuple $(r, P^{NM}, \vec{w}^{NM}, c)$, where r is a voting rule, P^{NM} is the non-manipulators' profile, $\vec{w}^{NM}$ represents the weights of P^{NM}, and c is the alternative preferred by the manipulators. We are asked to find the minimum W^{M} such that there exist a divisible vote V^{M} for one manipulator with weight W^{M}, such that*

$$c \in r((P^{NM}, \{V^{M}\}), (\vec{w}^{NM}, W^{M}))$$

We let WCMd, UCMd, UCOd denote the variants of WCM, UCM, UCO, respectively, in which votes are divisible. We note that it is irrelevant whether the votes of the non-manipulators are divisible or not; what matters is whether the manipulators' votes are divisible.

3. ALGORITHMS FOR WCM AND COD

In this section we present algorithms for WCM. For the divisible case, we devise a polynomial-time algorithm that solves WCMd by reducing it to the scheduling problem known as $Q|pmtn|C_{max}$. This algorithm also solves COd exactly. For WCM, we augment the algorithm for WCMd with a rounding technique, and obtain an approximation algorithm as a result. While our solution for WCMd may be interesting in its own right, its main purpose is to provide intuitions and techniques that are subsequently leveraged for approximating WCM.

[3]Our approach can be easily extended to the solve the unique-winner case, in which the objective is to find the infimum total weight of the manipulators that is sufficient to make c the unique winner.

3.1 The divisible case

We will show how to reduce WCMd/COd to the scheduling problem of *parallel uniform machines with preemption*, categorized as $Q|pmtn|C_{max}$ (see, for example, [3] for the meaning of the notation). In an instance of $Q|pmtn|C_{max}$, we are given n' jobs $\mathcal{J} = \{J_1, \ldots J_{n'}\}$ and m' machines $\mathcal{M} = \{M_1, \ldots, M_{m'}\}$; each job J_i has a workload $p_i \in \mathbb{R}_+$, and the processing speed of machine M_i is $s^i \in \mathbb{R}_+$, that is, it will finish s^i amount of work in one unit of time. A *preemption* is an interruption of the job that is being processed on one machine (the job may be resumed later, not necessarily on the same machine). Preemptions are allowed in $Q|pmtn|C_{max}$. We are asked for the minimum makespan, i.e., the minimum time to complete all jobs, and an optimal schedule.

We first draw a natural connection between WCMd/COd under positional scoring rules and $Q|pmtn|C_{max}$. After counting the non-manipulators' votes only, each alternative will have a total non-manipulator score. For any $i \leq m-1$, we let p_i denote the gap between the non-manipulator score of c_i and the non-manipulator score of c (which is positive if the former is larger; the case where the gap is negative is trivial). In particular, the p_i's can be seen as the workload of $m-1$ jobs. We note that, without loss of generality, the manipulators will always rank c in the top position. Therefore, a manipulator vote (of weight 1) in which c_j is ranked in the ith position decreases the gap between c_j and c by $s_1 - s_i$ points.

We consider a set of $m-1$ machines $M_1, \ldots, M_{m-1}$ whose speeds are $s_1 - s_2, \ldots, s_1 - s_m$, respectively. A ranking (a vote) is equivalent to an allocation of the $m-1$ jobs to machines: an alternative ranked i positions below c corresponds to a job allocated to the ith slowest machine. We can now see that the minimum makespan of the scheduling problem is the minimum total weight of the manipulators required to make c a winner, that is, the optimal solution to COd. For WCMd, the goal is to compute the votes for $\sum_{i=1}^{k} w_i$ "amount" of manipulators (since the votes are divisible, a problem instance with k manipulators with weights $\vec{w}$ is equivalent to a problem instance with a single manipulator whose weight is $\sum_{i=1}^{k} w_i$), such that the final total score of c is at least the final total score of any other alternative. This is equivalent to computing a schedule that completes all jobs within time at most $\sum_{i=1}^{k} w_i$.

Formally, for a WCMd instance $((s_1, \ldots, s_m), P^{NM}, w^{NM}, c, k, (w_1, \ldots, w_k))$, we construct an instance of $Q|pmtn|C_{max}$ with $m-1$ jobs and $m-1$ machines (that is, $m' = n' = m-1$) as follows. For any $i \leq m-1$, we let $s^i = s_1 - s_{i+1}$, $p_i = \max\{s(P^{NM}, w^{NM}, c_i) - s(P^{NM}, w^{NM}, c), 0\}$. We do not distinguish between alternative c_i and job J_i. This reduction is illustrated in the following example.

EXAMPLE 3.1. Let $m = 4$, $\mathcal{C} = \{c, c_1, c_2, c_3\}$. The positional scoring rule is Borda (which corresponds to the scoring vector $(3, 2, 1, 0)$). The non-manipulators are unweighted (that is, their weights are 1), and their profile is $P^{NM} = (V_1^{NM}, V_2^{NM}, V_3^{NM}, V_4^{NM})$, defined as follows.

$$V_1^{NM} = [c_1 \succ c \succ c_2 \succ c_3], \quad V_2^{NM} = [c_2 \succ c_1 \succ c \succ c_3]$$
$$V_3^{NM} = [c_3 \succ c_2 \succ c_1 \succ c], \quad V_4^{NM} = [c_1 \succ c_2 \succ c_3 \succ c]$$

We have that $s(P^{NM}, c) = 3$, $s(P^{NM}, c_1) = 9$, $s(P^{NM}, c_2) = 8$, $s(P^{NM}, c_3) = 4$. Therefore, we construct a $Q|pmtn|C_{max}$ instance in which there are 3 machines M_1, M_2, M_3 whose speeds are $s^1 = 1, s^2 = 2, s^3 = 3$, corresponding to the 2nd, 3rd, and 4th position in the votes, respectively, and 3 jobs J_1, J_2, J_3, whose workloads are $p_1 = 6 = (9-3), p_2 = 5 = (8-3), p_3 = 1 = (4-3)$, respectively. □

Let $W_0 = 0$, $W = \max_{j \leq k} w_j$, and for any $1 \leq i \leq k$, $W_i =$

$\sum_{j=1}^{i} w_j$. A schedule is usually represented by a *Gantt chart*, as illustrated in Figure 1. (We note that Figure 1 is not the solution to Example 3.1.)

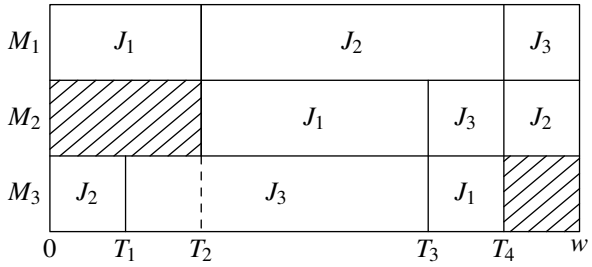

Figure 1: An example schedule. The machines are idle in shaded areas.

Let w be the minimum makespan for the $Q|pmtn|C_{max}$ instance constructed above, and let $f^* : \mathcal{M} \times [0, w] \to \mathcal{J} \cup \{I\}$ be an optimal solution to $Q|pmtn|C_{max}$, where I means that the machine is idle. If $w > W_k$, then there is no successful manipulation that makes c a winner. If $w \leq W_k$, we first extend the optimal solution f^* to make it fully occupy the whole time interval $[0, W_k]$; any way of allocating jobs to machines in the added time would suffice.[4] Let f be the solution obtained in this way.

Given f, for any time $t \in [0, W_k]$, we say that t is a *preemptive break point* if there is a preemption at t—formally, there exists a machine M_i such that for some $\epsilon' > 0$, we have that for all $\epsilon \in [0, \epsilon']$, $f(M_i, t-\epsilon) \neq f(M_i, t+\epsilon)$, that is, the job being processed at time $t - \epsilon$ on M_i is different from the job being processed at time $t+\epsilon$. We let $B_f = \{T_1, \ldots, T_l\}$ denote the preemptive break points of f, where $0 < T_1 < T_2 < \ldots < T_l < W_k$. For example, the set of preemptive break points of the schedule in Figure 1 is $B_f = \{T_1, T_2, T_3, T_4\}$.

EXAMPLE 3.2. The minimum makespan of the scheduling problem instance in Example 3.1 is $(6 + 5)/5 = 11/5$. An optimal schedule f is as follows.

$\mathbf{M_1}$: For any $0 \leq t \leq 11/5$, $f(M_1, t) = J_3$.

$\mathbf{M_2}$: For any $0 \leq t \leq 8/5$, $f(M_2, t) = J_2$; for any $8/5 < t \leq 11/5$, $f(M_2, t) = J_1$.

$\mathbf{M_3}$: For any $0 \leq t \leq 8/5$, $f(M_3, t) = J_1$; for any $8/5 < t \leq 11/5$, $f(M_3, t) = J_2$.

$t = 8/5$ is the only preemptive break point in this schedule. □

Any solution to the $Q|pmtn|C_{max}$ instance obtained from the reduction can be converted to a solution to WCMd in the following way. First, we assign jobs to all idle machines arbitrarily to ensure that at any time between 0 and W_k, no machines are idle and all jobs are allocated. Formally, we define $f' : \mathcal{M} \times [0, W_k] \to \mathcal{J}$ such that $\{f'(M_1, t), \ldots, f'(M_{m-1}, t)\} = \{J_1, \ldots, J_{m-1}\}$ for all t, and for any $M \in \mathcal{M}$ and $t \in [0, W_k]$, we have that if $f(M, t) \in \mathcal{J}$, then $f'(M, t) = f(M, t)$. For example, we can assign jobs to the shaded areas (which represent idle time) in the schedule in Figure 1 in the way illustrated in Figure 2.

Next, for any $1 \leq i \leq k$, we convert the schedule to the manipulators' votes in the natural way:

[4]This works for the co-winner case. For the unique-winner case, in order to have a solution we need $w < W_k$, and then in the time interval $[w, W_k]$ we allocate the jobs in an arbitrary way such that each job runs on each machine for some time.

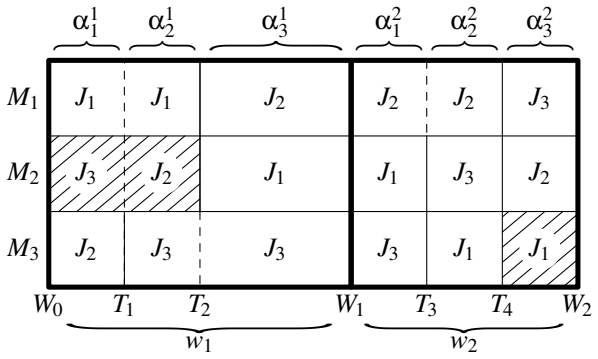

Figure 2: Conversion of an optimal schedule to a solution for WCMd.

- If there are no preemption break points in (W_{i-1}, W_i), we let manipulator i vote for $c \succ f'(M_1, W_{i-1}+\epsilon) \succ f'(M_2, W_{i-1}+\epsilon) \succ \ldots \succ f'(M_{m-1}, W_{i-1} + \epsilon)$, where $\epsilon > 0$ is sufficiently small.
- If there are preemptive break points in (W_{i-1}, W_i), denoted by $T_a, T_{a+1}, \ldots, T_{a+b-1}$, then we let $V_1^i, \ldots, V_{b+1}^i$ denote the orders that correspond to the schedule at times $W_{i-1} + \epsilon, T_a + \epsilon, \ldots, T_{a+b-1} + \epsilon$, respectively. Let $\alpha_1^i = T_a - W_{i-1}$, $\alpha_2^i = T_{a+1} - T_a, \ldots, \alpha_{b+1}^i = W_i - T_{a+b-1}$. We let manipulator i vote for $\sum_{j=1}^{b+1}[\alpha_j^i/(W_i - W_{i-1})] \cdot V_j^i$.

EXAMPLE 3.3. Suppose there are two manipulators whose weights w_1 and w_2 are illustrated in Figure 2. Manipulator 1 votes $[(1/4)(c \succ c_1 \succ c_3 \succ c_2) + (1/4)(c \succ c_1 \succ c_2 \succ c_3) + (1/2)(c \succ c_2 \succ c_1 \succ c_3)]$; manipulator 2 votes $[(1/3)(c \succ c_2 \succ c_1 \succ c_3) + (1/3)(c \succ c_2 \succ c_3 \succ c_1) + (1/3)(c \succ c_3 \succ c_2 \succ c_1)]$. □

On the basis of the exposition above we now refer the reader to Algorithm 1. The algorithm solves WCMd in three steps: 1. convert the WCMd instance to a $Q|pmtn|C_{max}$ instance; 2. apply a polynomial-time algorithm that solves $Q|pmtn|C_{max}$ (for example, the algorithm in [13]); 3. convert the solution to the scheduling instance to a solution to the WCMd instance. Algorithm 1 also solves COd, because the makespan w computed in Line 3 is the optimal solution to COd. It is easy to verify that the algorithm runs in polynomial time. To conclude, we have the following result.

THEOREM 3.4. *Algorithm 1 solves WCMd and COd (exactly) in polynomial time.*

3.2 The indivisible case

We now move on to the more difficult indivisible case. We first note that Algorithm 1 cannot be directly applied to WCM, because the manipulators' votes constructed in Line 16 can be divisible. For any positional scoring rule, if there is a successful manipulation (in which all manipulators rank c in the top position), and we increase the weights of the manipulators, then c still wins the election. This property is known as *monotonicity in weights* (see [23] for a formal definition and the proof). Therefore, instead of having manipulator i cast the divisible vote $\sum_j[\alpha_j^i/(W_i - W_{i-1})] \cdot V_j^i$, we let her cast the indivisible vote $V_{j^*}^i$, which is one of the V_j^i with the highest weight among all the V_j^i's constructed for manipulator i. In addition, for any $j \neq j^*$, we add one extra manipulator whose weight is α_j^i, and let the new manipulator vote V_j^i. It turns out that if we use a particular algorithm for the scheduling problem, then the solution will not require too many additional manipulators. This gives us Algorithm 2 for WCM.

Algorithm 1: compWCMd

1 $\forall i \leq m-1$, $s^i \leftarrow s_1 - s_{i+1}$
2 $\forall i \leq m-1$,
$p_i \leftarrow \max\{s(P^{NM}, w^{NM}, c_i) - s(P^{NM}, w^{NM}, c), 0\}$
3 Solve the $Q|pmtn|C_{max}$ instance (for example, using the algorithm in [13]). Let w and f denote the minimum makespan and an extended optimal schedule; let $T_1, \ldots, T_l$ denote the preemptive break points.
4 **if** $w > W_k$ **then**
5 **return** *false*.
6 **end**
7 Let $f' : \mathcal{M} \times [0, W_k] \rightarrow \mathcal{J}$ be such that $\{f'(M_1, t), \ldots, f'(M_{m-1}, t)\} = \{J_1, \ldots, J_{m-1}\}$, and for any $M \in \mathcal{M}$, any $t \in [0, W_k]$, we have that if $f(M, t) \in \mathcal{J}$, then $f'(M, t) = f(M, t)$.
8 **for** $i = 1$ **to** k **do**
9 Let $V_1^i = [c \succ f'(M_1, W_{i-1} + \epsilon) \succ \ldots \succ f'(M_{m-1}, W_{i-1} + \epsilon)]$
10 $j \leftarrow 2$
11 **for** *each preemptive break point* $T \in (W_{i-1}, W_i)$ *(in order)* **do**
12 Let $V_j^i = [c \succ f'(M_1, T+\epsilon) \succ \ldots \succ f'(M_{m-1}, T+\epsilon)]$
13 $j \leftarrow j + 1$
14 **end**
15 For any j, let α_j^i be the length of the jth interval in $[W_{i-1}, W_i]$ induced by the preemptive break points.
16 Let manipulator i vote $\sum_j [\alpha_j^i/(W_i - W_{i-1})] \cdot V_j^i$, and add this vote to P^M
17 **end**
18 **return** P^M

Algorithm 2: compWCM

This algorithm is the same as Algorithm 1, except for the following two lines:
3 Use the algorithm in [13] to solve the scheduling problem
16 Let manipulator i vote for $V_{j^*}^i$, where for any $j \neq j^*$, $\alpha_{j^*}^i \geq \alpha_j^i$; and for any $j \neq j^*$, we add a new manipulator whose weight is α_j^i, and let her vote V_j^i

EXAMPLE 3.5. Let the coalitional manipulation problem instance be the same as in Example 3.1. Suppose we have two manipulators whose weights are both 1; then, because the minimum makespan is $11/5 > 2$ (as observed in Example 3.2), there is no solution to the WCMd and WCM problem instances. The solution to the COd problem instance is $11/5$.

Now suppose we have two manipulators, whose weights are $w_1 = 1$ and $w_2 = 6/5$, respectively. Let f be the optimal schedule defined in Example 3.2. A solution to the WCMd problem instance is obtained as follows. Manipulator 1 votes $[c \succ c_3 \succ c_2 \succ c_1]$, and manipulator 2 votes $[(1/2)(c \succ c_3 \succ c_2 \succ c_1) + (1/2)(c \succ c_3 \succ c_1 \succ c_2)]$. For WCM, the vote of manipulator 1 is the same, the vote of manipulator 2 is $[c \succ c_3 \succ c_2 \succ c_1]$, and there is one additional manipulator, whose weight is $3/5$ and whose vote is $[c \succ c_3 \succ c_1 \succ c_2]$. □

EXAMPLE 3.6. Suppose there are two manipulators whose weights are illustrated in Figure 2. The vote of manipulator 1 is $c \succ c_2 \succ c_1 \succ c_3$, and we introduce two new manipulators with weight $w_1/4$ whose votes are $c \succ c_1 \succ c_3 \succ c_2$ and $c \succ c_1 \succ c_2 \succ c_3$. The vote of manipulator 2 is $c \succ c_2 \succ c_1 \succ c_3$, and we introduce two new manipulators with weight $w_2/3$ whose votes are $c \succ c_2 \succ c_3 \succ c_1$ and $c \succ c_3 \succ c_2 \succ c_1$. Since $|B_f|$ (the number of preemptive break points) is 4, there are in total four additional manipulators. □

For any $j \neq j^*$, we must have $\alpha_j^i \leq (W_i - W_{i-1})/2 \leq W/2$ (recall that $W = \max_{j \leq k} w_j$). Moreover, for any preemptive break point we introduce at most one extra manipulator. Therefore, we immediately have the following lemma that relates the number of the new manipulators to the number of preemptive break points.

LEMMA 3.7. *If $w \geq W_k$, then there is no successful manipulation for WCMd (nor for WCM); otherwise, Algorithm 2 returns a manipulation with at most $|B_f|$ additional manipulators, each with weight at most $W/2$.*

Therefore, the smaller $|B_f|$ is, the fewer new manipulators are introduced by Algorithm 2. $|B_f|$ depends on which algorithm we use to solve $Q|pmtn|C_{max}$ in Line 3. In fact, there are many efficient algorithms that solve $Q|pmtn|C_{max}$. For example, $Q|pmtn|C_{max}$ can be solved in time $O(n'^2 m')$ by a greedy algorithm [3]. At each time point t, the algorithm (called the *level algorithm*) assigns jobs to the machines in a way such that the greater the remaining workload of a job, the faster the machine it is assigned to.[5] However, this algorithm in some cases generates a schedule that has as many as $m'(m'-1)/2$ preemptive break points. Therefore, we turn to the algorithm by Gonzalez and Sahni [13], which runs in time $O(n' + m' \log n')$ using at most $2(m'-1)$ preemptions. Gonzalez and Sahni also showed that this bound is tight. We note that one preemptive break point corresponds to at least two preemptions, and in the instances that were used to show that the $2(m'-1)$ bound is tight, $m'-1$ preemptive break points are required. Therefore, we immediately have the following lemma.

LEMMA 3.8. *The number of preemptive break points in the solution obtained by the algorithm of Gonzalez and Sahni [13] is at most $m'-1$. Furthermore, this bound is tight.*

We note that $m' = m - 1$. Hence, combining Lemma 3.7 and Lemma 3.8, we have the following theorem, which is our main result.

THEOREM 3.9. *Algorithm 2 runs in polynomial time and*

1. *if the algorithm returns* false*, then there is no successful manipulation (even for the WCMd version of the instance);*
2. *otherwise, the algorithm returns a successful manipulation for a modified set of manipulators, consisting of the original manipulators plus at most $m-2$ additional manipulators, each with weight at most $W/2$.*

4. ALGORITHMS FOR UCM AND UCO

We now consider the case where votes are unweighted. UCMd and UCOd can be solved using Algorithm 1. As for UCM/UCO, every manipulator's weight is one (so that $W = 1$), and we are only allowed to add new manipulators whose weight is also 1. We recall that increasing the weights of the manipulators never prevents c from winning. Therefore, in the context of UCM/UCO we use a slight modification of Algorithm 2, by adding one unweighted manipulator whenever Algorithm 2 proposes adding a weighted manipulator (whose weight can be at most $1/2$).

[5]The greedy algorithm of Zuckerman et al. [23] is effectively a discrete-time version of the level algorithm.

Algorithm 3: compWCM

This algorithm is the same as Algorithm 1, except for the following two lines:

3 Use the algorithm in [13] to solve the scheduling problem.

16 Let manipulator i vote for V_1^i; for any $j > 1$, we add a new manipulator who votes for V_j^i.

The following corollary immediately follows from Theorem 3.9.

COROLLARY 4.1. *For UCM, if Algorithm 3 returns* false*, then there is no successful manipulation; otherwise, Algorithm 3 returns a successful manipulation with at most $m-2$ additional manipulators.*

Recall that Lines 1-3 of Algorithm 3 compute the minimum makespan w (the solution to COd) of the scheduling problem that is obtained from the UCM instance. It is easy to see that if votes are divisible then $\lceil w \rceil$ is the minimum number of unweighted manipulators required to make c win the election, that is, $\lceil w \rceil$ is the optimal solution to UCOd. Therefore, Algorithm 1 can easily be modified to yield an algorithm that solves UCOd. We further note that Algorithm 3 is an approximation algorithm for UCO, as the number of manipulators returned by Algorithm 3 is no more than $\lceil w \rceil + m - 2$. Put another way, Algorithm 3 returns a solution to UCO (with indivisible votes) that approximates the optimal solution to UCOd (with divisible votes) to an additive term of $m-2$.

Generally, if there exists a successful manipulation, then Algorithm 3 returns a manipulation with additional manipulators. However, there are some special positional scoring voting rules under which UCM can always be solved exactly by Algorithm 1. Given $k \in \{1, \ldots, m-1\}$, the k-approval rule is the scoring rule where $s_1 = \ldots = s_k = 1$ and $s_{k+1} = \ldots = s_m = 0$. For example, Plurality (with scoring vector $(1, 0, \ldots, 0)$) and Veto (with scoring vector $(1, \ldots, 1, 0)$) are 1-approval and $(m-1)$-approval, respectively. We note that UCM under any k-approval rule reduces to the scheduling problem in which all machines have the same speed. This corresponds exactly to the scheduling problem $P|pmtn|C_{max}$ in discrete time (that is, the preemptions are allowed only at integer time points), which has a polynomial-time algorithm: *Longest Remaining Processing Time first (LRPT)* [16]. Therefore, if we modify Algorithm 3 by solving the reduced scheduling instance with LRPT, then we can solve UCM under any k-approval voting rule in polynomial time.[6] To summarize:

COROLLARY 4.2. *Let $k \in \{1, \ldots, m-1\}$. UCM/UCO under k-approval is in P.*

4.1 On the tightness of the results

We presently wish to argue that we have made the most of our technique. The next theorem states that the $m-2$ bound is tight in terms of the difference between the optimal solution to UCO and the optimal solution to UCOd under the same input. It also implies that Algorithm 3 is optimal in the sense that for any $q < m-2$, there is no approximation algorithm for UCO that always outputs a manipulation with at most q manipulators more than the optimal solution to UCOd. This result can be seen as a new type of integrality gap, which applies to our special flavor of rounding.

[6]The simple observation that UCM is in P under approval voting rules was also recently made by Andrew Lin (via personal communication), who employed a completely different (greedy) approach.

THEOREM 4.3. *For any $m \geq 3$, there exists a UCO instance such that the (additive) gap between the optimal solution to UCOd and the optimal solution to UCO is $m-2$.*

PROOF. For any $m \geq 3$, we let the scoring vector be $(m(m-1)(m-2)-1, \ldots, m(m-1)(m-2)-1, m(m-1)(m-2)-2, 0)$. Let $V = [c_1 \succ \ldots \succ c_{m-1} \succ c]$, and let π be the cyclic permutation on $\mathcal{C} \setminus \{c\}$, that is, $\pi : c_1 \to \ldots \to c_{m-1} \to c_1$. For any $i \leq m-1$, let V_i be the linear order over $\mathcal{C}$ in which c is ranked in position $(m-1)$, and $\pi^i(c_1) \succ_{V_i} \pi^i(c_2) \succ_{V_i} \ldots \succ_{V_i} \pi^i(c_{m-1})$. Let $P = (V, V_1, \ldots, V_{m-1})$, $P^{NM} = P \cup \pi(P) \cup \ldots \cup \pi^{m-2}(P)$. It follows that for any $i \leq m-1$, $s(P^{NM}, c_i) - s(P^{NM}, c) = (m-1)^2 - 1$. Let $V' = [c \succ c_1 \ldots \succ c_{m-1}]$; it can be verified that the divisible vote

$$\frac{1}{m-1}(V', \pi(V'), \pi^2(V'), \ldots, \pi^{m-2}(V'))$$

is sufficient to make c win, hence the optimal solution to UCOd is 1.

We next prove that the solution to UCO is $m-1$. Clearly the profile $(V', \pi(V'), \pi^2(V'), \ldots, \pi^{m-1}(V'))$ is a successful manipulation. Hence, it remains to show that the solution is at least $m-1$. For the sake of contradiction we assume that the solution is $m-2$, and P^M is the corresponding successful manipulation. Therefore, there must exist $i \leq m-1$ such that c_i is not ranked at the bottom of any of the votes of P^M. Therefore,

$$s(P^M, c) - s(P^M, c_i) \leq m-2 < (m-1)^2 - 1,$$

which means that $s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, c_i) < 0$. This contradicts the assumption that P^M is a successful manipulation. □

We next ask the following natural question: is it possible to improve the rounding technique so that the algorithm achieves a better bound, relative to the optimal solution for the indivisible case? This is not ruled out by Theorem 4.3, since that theorem compares to the optimal UCOd solution rather than the optimal UCO solution. Nevertheless, the answer is negative, as long as all linear orders in an optimal solution to the COd problem appear in the output of the algorithm. We say that an approximation algorithm $\mathcal{A}$ for UCO is *based on COd* if for any UCO instance, there exists an optimal solution to COd such that every linear order that appears in that solution also appears in the output of $\mathcal{A}$ (as a fraction of the vote of a manipulator).

THEOREM 4.4. *Let $\mathcal{A}$ be an approximation algorithm based on COd. For any $m \geq 3$, there exists a UCO instance such that the gap between the optimal solution to UCO and the output of $\mathcal{A}$ is $m-2$.*

PROOF. For any $m \geq 3$, we construct an instance such that the solution to the UCO problem is 1, but at least $m-1$ linear orders appear in any optimal solution to the COd problem (so the gap is $m-2$).

We let the scoring vector be $(m+2, 1, 0, \ldots, 0)$. Let

$$V = [c \succ c_1 \succ \ldots \succ c_{m-1}],$$

and

$$V' = [c_{m-1} \succ c_1 \succ c \succ c_2 \succ \ldots \succ c_{m-2}].$$

Furthermore, let

$$\pi : c_1 \to c_2 \to \ldots \to c_{m-1} \to c_1,$$

and

$$\pi^* : c \to c_1 \to \ldots \to c_{m-1} \to c.$$

We define preference profiles by letting

$$P = (V', V, \pi^*(V), (\pi^*)^2(V), \ldots, (\pi^*)^{m-2}(V))$$

$$\text{and } P^{NM} = P \cup \pi(P) \cup \ldots \cup \pi^{m-2}(P).$$

We have that $s(P, c) = m + 2$, $s(P, c_1) = m + 4$, and for any $2 \leq i \leq m - 1$, $s(P, c_i) = m + 3$. Therefore, $s(P^{NM}, c) = (m + 2)(m - 1)$ and for any $2 \leq i \leq m - 1$, $s(P^{NM}, c_i) = (m+3)(m-1)+1$. Therefore, for any $i \leq m-1$, $s(P^{NM}, c_i) - s(P^{NM}, c) = m$. It follows that one manipulator suffices to make c the winner (by voting $c \succ c_1 \succ \ldots \succ c_{m-1}$).

On the other hand, the minimum weight for COd is $(m - 1)/m$, for example,

$$V^M = \frac{m-1}{m}(\frac{1}{m-1}V + \frac{1}{m-1}\pi(V) + \ldots + \frac{1}{m-1}\pi^{m-2}(V)).$$

In any manipulator's vote corresponding to the minimum total weight, every alternative except c must appear in the second position for a fraction of the vote. Therefore, any algorithm based on COd must output at least $m - 1$ linear orders. □

5. UCM UNDER POSITIONAL SCORING RULES IS STRONGLY NP-COMPLETE

In this section, we show that UCM under a specific positional scoring rule is strongly NP-complete, even when there are only two manipulators. We slightly abuse terminology here, since a voting rule is formally defined with respect to a specific number of alternatives; for the purposes of this section, a positional scoring rule defines a separate score vector for each possible number of alternatives. Indeed, Plurality, Veto, and Borda fit this description, so the rule that we introduce here is a single positional scoring rule in the same sense that these three rules are.

Let us define our positional scoring rule, denoted by r_{weird}. Given $K \in \mathbb{N}$, the scoring vector for $8K^2 + 1$ alternatives is

$$(10K, \underbrace{10K-1, \ldots, 10K-1}_{2K}, \underbrace{10K-2, \ldots, 10K-2}_{2K}, \ldots,$$
$$\underbrace{9K, \ldots, 9K}_{2K}, \underbrace{7K, \ldots, 7K}_{2K^2}, \underbrace{3K, \ldots, 3K}_{2K^2}, \underbrace{K, \ldots, K}_{2K},$$
$$\underbrace{K-1, \ldots, K-1}_{2K}, \ldots, \underbrace{1, \ldots, 1}_{2K})$$

If the number of alternatives m cannot be written as $8K^2 + 1$ for some K, our scoring rule can behave arbitrarily.

We have the following theorem, whose proof appears in Appendix A.

THEOREM 5.1. *UCM under r_{weird} is strongly NP-complete, even when the number of manipulators is two.*

It follows from the proof that $Q|pmtn|C_{max}$ is strongly NP-complete in discrete time.

Acknowledgments

We thank Edith Hemaspaandra, Lane Hemaspaandra, Jeff Rosenschein, and Michael Zuckerman for helpful comments. Lirong Xia is supported by a James B. Duke Fellowship and Vincent Conitzer is supported by an Alfred P. Sloan Research Fellowship. Xia and Conitzer are also supported by NSF under award numbers IIS-0812113 and CAREER 0953756. Ariel Procaccia is supported by a Rothschild Postdoctoral Fellowship.

6. REFERENCES

[1] John Bartholdi, III and James Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[2] John Bartholdi, III, Craig Tovey, and Michael Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[3] Peter Brucker. *Scheduling Algorithms*. Springer Publishing Company, Incorporated, 2007.

[4] Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. of IJCAI-03*, pages 781–788, 2003.

[5] Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proc. of AAAI-06*, pages 627–634, 2006.

[6] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[7] Shahar Dobzinski and Ariel D. Procaccia. Frequent manipulability of elections: The case of two voters. In *Proc. of WINE-08*, pages 653–664, 2008.

[8] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proc. of ISAAC-05*, pages 206–215, 2005.

[9] Piotr Faliszewski, Edith Hemaspaandra, and Henning Schnoor. Copeland voting: ties matter. In *Proc. of AAMAS-08*, pages 983–990, 2008.

[10] Ehud Friedgut, Gil Kalai, and Noam Nisan. Elections can be manipulated often. In *Proc. of FOCS-08*, pages 243–249, 2008.

[11] Michael Garey and David Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[12] Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41:587–602, 1973.

[13] Teofilo Gonzalez and Sartaj Sahni. Preemptive scheduling of uniform processor systems. *J. ACM*, 25(1):92–101, 1978.

[14] Edith Hemaspaandra and Lane A. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

[15] Noam Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 9. Cambridge University Press, 2007.

[16] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.

[17] Ariel D. Procaccia and Jeffrey S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, 2007.

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

[19] Lirong Xia and Vincent Conitzer. Generalized scoring rules and the frequency of coalitional manipulability. In *Proc. of EC-08*, pages 109–118, 2008.

[20] Lirong Xia and Vincent Conitzer. A sufficient condition for voting rules to be frequently manipulable. In *Proc. of EC-08*, pages 99–108, 2008.

[21] Lirong Xia and Vincent Conitzer. Finite local consistency

characterizes generalized scoring rules. In *Proc. of IJCAI-09*, pages 336–341, 2009.

[22] Lirong Xia, Michael Zuckerman, Ariel D. Procaccia, Vincent Conitzer, and Jeffrey S. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proc. of IJCAI-09*, pages 348–353, 2009.

[23] Michael Zuckerman, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009. Preliminary version in SODA-08.

APPENDIX

A. PROOF OF THEOREM 5.1

We prove the hardness by a reduction from NUMERICAL MATCHING WITH TARGET SUMS (NMTS), which is strongly NP-complete [11]. An NMTS instance consists of three disjoint sets A, B, Y where $|A| = |B| = |Y| = l \geq 2$, and a weight function $w : A \cup B \cup Y \to \mathbb{N}$. We are asked whether there is a partition $S = S_1 \cup \ldots \cup S_l$ of $A \cup B \cup Y$ such that for any $i \leq l$, $S_i = \{a^i, b^i, y^i\}$, where $a^i \in A, b^i \in B, y^i \in Y$ and $w(a^i) + w(b^i) = w(y^i)$.

Let A, B, Y, w be an NMTS instance, where $A = \{a_1, \ldots, a_l\}$, $B = \{b_1, \ldots, b_l\}$, $Y = \{y_1, \ldots, y_l\}$, $w(a_1) \leq w(a_2) \leq \ldots \leq w(a_l)$, $w(b_1) \leq w(b_2) \leq \ldots \leq w(b_l)$. W.l.o.g., we make the following assumption about the NMTS instance.

ASSUMPTION 1.

- *For any $a, a', a^* \in A$, and any $b \in B$, we have $w(a^*) < w(a) + w(a') < w(b)$.*
- *For any $a \in A$, and any $b, b' \in B$, we have $w(a) < w(b') < w(a) + w(b)$.*
- *For any $b, b' \in B$, and any $y \in Y$, we have $w(b) < w(y) < w(b) + w(b')$.*

This assumption does not limit generality since we can convert any instance A, B, Y, w' to an NMTS instance A, B, Y, w whose inputs are polynomially larger (in the unary sense) in the following way: let

$$w'_{max} = \max_{a \in A, b \in B, y \in Y} \{w'(a), w'(b), w'(y)\}$$

For any $a \in A, b \in B, y \in Y$, let $w(a) = w'(a) + 2(w'_{max} + 1)$, $w(b) = w'(b) + 6(w'_{max} + 1)$, and $w(y) = w'(y) + 8(w'_{max} + 1)$.

Given an NMTS instance that satisfies Assumption 1, we construct the UCM instance as follows. The manipulators' goal is to make c a co-winner. A similar reduction exists for the unique-winner case. Let $K = \max\{w_{max}, l\}$, where

$$w_{max} = \max_{a \in A, b \in B, y \in Y} \{w(a), w(b), w(y)\}.$$

Alternatives: There are $8K^2 + 1$ alternatives. $\mathcal{C} = \{c\} \cup Y \cup D_A \cup D_B \cup D$, where $D_A = \{d_1^A, \ldots d_{2K^2-l}^A\}$, $D_B = \{d_1^B, \ldots d_{2K^2-l}^B\}$, $D = \{d_1, \ldots, d_{4K^2+l}\}$.

Non-manipulators' profile: $P^{NM} = P_1 \cup P_2$. We first describe the properties that P_1 and P_2 satisfy, then show how to construct them.

- P_1 satisfies the following condition: Let F be a multiset, defined as

$$F = \{\underbrace{1, \ldots, 1}_{2K}, \underbrace{2, \ldots, 2}_{2K}, \ldots, \underbrace{K, \ldots, K}_{2K}\}.$$

  That is, F is composed of $2K$ copies of $\{1, 2, \ldots, K\}$. Let $E = E_A \cup E_B$, where $E_A = F \setminus w(A)$ and $E_B = F \setminus w(B)$, where $w(A)$ is a multiset, defined as $w(A) = \{w(a) : a \in A\}$ (similarly for $w(B)$). We also write $E_A = \{e_1^A, \ldots, e_{2K^2-l}^A\}$; $E_B = \{e_1^B, \ldots, e_{2K^2-l}^B\}$.

  For any $i \leq 2K^2 - l$, we have $s(P_1, c) - s(P_1, d_i^A) = e_i^A - 17K$ and $s(P_1, c) - s(P_1, d_i^B) = e_i^B - 17K$; for any $1 \leq j \leq l$, we let $s(P_1, c) - s(P_1, y_j) = w(y_j) - 20K$.
- P_2 satisfies the following conditions: for any $x \in Y \cup D_A \cup D_B$, we have $s(P_2, c) = s(P_2, x)$; for any $i \leq 4K^2 + l$, we have $s(P_1 \cup P_2, c) > s(P_1 \cup P_2, d_i)$.

To construct P_1, we first make the following observation: for any $\{x_1, \ldots, x_L\} = X \subseteq \mathcal{C}$ where $L \leq 4K^2$, and any $x \in \mathcal{C} \setminus X$, let

$$V_1 = [x_1 \succ x \succ x_2 \succ \ldots \succ x_L \succ (\mathcal{C} \setminus (X \cup \{x\}))]$$

and

$$V_2 = [(\mathcal{C} \setminus X) \succ x_L \succ x_{L-1} \succ \ldots \succ x_1],$$

where the elements in $\mathcal{C} \setminus X$ are ranked in an arbitrary way; letting $P^* = (V_1, V_2)$, we must have that for any $2 \leq i \leq L$, $s(P^*, x_1) - s(P^*, x_i) = 1$. Therefore, P_1 can be constructed out of no more than $2 \cdot 20K \cdot (4K^2 - l)$ votes (by choosing $X = \{c\} \cup Y \cup D_A \cup D_B$, and applying the P^* trick no more than $20K$ times per alternative in X), and for any $d \in D$, we have $s(P_1, c) - s(P_1, d) \geq -2 \cdot 20K \cdot (4K^2 - l) \cdot 10K$.

Next we show how to construct P_2. Let π_1 be the cyclic permutation on $\{c\} \cup Y \cup D_A \cup D_B$, defined as $c \to y_1 \to \ldots \to y_l \to d_1^A \to \ldots \to d_{2K^2-l}^A \to d_1^B \to \ldots \to d_{2K^2-l}^B \to c$. Let π_2 be the cyclic permutation on D, defined as $d_1 \to d_2 \to \ldots \to d_{4K^2+l} \to d_1$.

For any $t \in \mathbb{N}$, we let $\pi_1^t = \pi_1 \circ \pi_1^{t-1}$, $\pi_1^1 = \pi_1$, where for any $x \in \{c\} \cup Y \cup D_A \cup D_B$, $\pi_1 \circ \pi_1^{t-1}(x) = \pi_1(\pi_1^{t-1}(x))$. π_2^t is defined similarly. We note that $\pi_1^{4K^2-l+2} = \pi_1$, $\pi_2^{4K^2+l+1} = \pi_2$. For any $j \in \mathbb{N}$, we let

$$\begin{aligned} W_j = &[(\pi_1^j(c) \succ \pi_1^j(y_1) \succ \ldots \succ \pi_1^j(y_l) \succ \pi_1^j(d_1^A) \succ \ldots \\ &\succ \pi_1^j(d_{2K^2-l}^A) \succ \pi_1^j(d_1^B) \succ \ldots \succ \pi_1^j(d_{2K^2-l}^B) \\ &\succ \pi_2^j(d_1) \succ \ldots \succ \pi_2^j(d_{4K^2+l})] \end{aligned}$$

Let $P' = (W_1, \ldots, W_{(4K^2-l+1)(4K^2+l)})$. It follows that for any $x \in Y \cup D_A \cup D_B$, we have $s(P', c) = s(P', x)$; for any $d \in D$, we have

$$\begin{aligned} &(P', c) - s(P', d) \\ >&(4K^2 + l) \cdot 7K \cdot (4K^2 - l + 1) \\ &- (4K^2 - l + 1)(4K^2 \cdot 3K + 7K \cdot l) \\ =&(4K^2 - l + 1)16K^3 \end{aligned}$$

Let P_2 be composed of 25 copies of P'. For any $d \in D$, we have

$$\begin{aligned} &s(P_1 \cup P_2, c) - s(P_1 \cup P_2, d) \\ =&s(P_1, c) - s(P_1, d) + s(P_2, c) - s(P_2, d) \\ >&-400K^2(4K^2 - l) + 25 \cdot 16K^2(4K^2 - l + 1) \\ >&0. \end{aligned}$$

This completes the description of the reduction.

Next, we show that the UCM instance has a solution if and only if the NMTS instance has a solution. Assume that the NMTS problem has a solution $S_1, \ldots, S_l$. W.l.o.g., for any $i \leq l$, $S_i = \{a_{\pi(i)}, b_{\gamma(i)}, y_i\}$, where π and γ are permutations over $\{1, \ldots l\}$. We construct two votes Q_1, Q_2 that satisfy the following conditions.

- For any $i \leq 2K^2 - l$, we have $s(\{Q_1\}, d_i^A) = e_i^A$ and $s(\{Q_1\}, d_i^B) = 3K$; for any $j \leq l$, we have $s(\{Q_1\}, y_j) = w(a_{\pi(j)})$.
- For any $i \leq 2K^2 - l$, we have $s(\{Q_2\}, d_i^A) = 3K$, $s(\{Q_2\}, d_i^B) = e_i^B$; for any $j \leq l$, we have $s(\{Q_2\}, y_j) = w(b_{\gamma(j)})$.

In Q_1 and Q_2, c is ranked in the top position, and the alternatives in D are ranked arbitrarily. Q_1 and Q_2 are well defined, because $F = E_A \cup w(A) = E_B \cup w(B)$. Let $P^M = (Q_1, Q_2)$. For any $j \leq l$, we have the following calculations. First,

$$\begin{aligned} &s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, y_j) \\ =&w(y_j) - 20K + 20K - (w(a_{\pi(j)}) + w(b_{\gamma(j)})) = 0 \end{aligned}$$

For any $i \leq 2K^2 - l$, we have

$$\begin{aligned} &s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, d_i^A) \\ =&e_i^A - 17K + 20K - (e_i^A + 3K) = 0, \end{aligned}$$

and

$$\begin{aligned} &s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, d_i^B) \\ =&e_i^B - 17K + 20K - (e_i^B + 3K) = 0. \end{aligned}$$

For any $1 \leq i \leq 4K^2 + l$, we have

$$s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, d_i) > 0 + 2 > 0.$$

Therefore, c is a co-winner of the election.

Finally, we prove that if the UCM instance has a solution $P^M = (Q_1, Q_2)$, then the NMTS instance has a solution. First we note that for any $1 \leq i \leq 2K^2 - l$, d_i^A must be ranked within $4K^2$ positions from the bottom in both Q_1 and Q_2, otherwise d_i^A will obtain at least $7K$ points in P^M, thus

$$\begin{aligned} &s(P^{NM} \cup P^M, c) - s(P^{NM} \cup P^M, d_i^A) \\ &\leq K - 17K + 20K - 7K < 0, \end{aligned}$$

which means that c does not win the election. Similarly, any alternative in D_B and Y must be ranked within $4K^2$ positions from the bottom in both Q_1 and Q_2.

It is easy to check that for any $1 \leq i \leq 2K^2 - l$, we must have that $s(P^M, d_i^A) = 3K + e_i^A$ and $s(P^M, d_i^B) = 3K + e_i^B$; for any $y \in Y$, we must have that $s(P^M, y) = w(y)$.

Next, for any $1 \leq i \leq 2K^2 - l$, we must have that

$$\{s(Q_1, d_i^A), s(Q_2, d_i^A)\} = \{3K, e_i^A\}$$

and

$$\{s(Q_1, d_i^B), s(Q_2, d_i^B)\} = \{3K, e_i^B\},$$

because $e_i^A \leq K$ and $e_i^B \leq K$. For any $y \in Y$, we have $3K \notin \{s(Q_1, y), s(Q_2, y)\}$, because $3K > w(y)$. Hence,

$$\begin{aligned} &\{s(Q_1, y), s(Q_2, y) : y \in Y\} \\ =&\{w(a_1), \ldots, w(a_l), w(b_1), \ldots, w(b_l)\} \end{aligned}$$

Note that both sides are multisets. We further note that for any $y \in Y$, any $a, a' \in A$, and any $b, b' \in B$, we have $w(a) + w(a') < w(y) < w(b) + w(b')$ and $w(a) < w(b)$ (Assumption 1). Therefore, $\{\min(s(Q_1, y), s(Q_2, y)) : y \in Y\} = w(A)$ and $\{\max(s(Q_1, y), s(Q_2, y)) : y \in Y\} = w(B)$. Let $f_A : Y \to A$ be a bijection such that for any $y \in Y$, $\min(s(Q_1, x), s(Q_2, x)) = w(f_A(y))$; let $f_B : Y \to B$ be a bijection such that for any $y \in Y$, $\max(s(Q_1, x), s(Q_2, x)) = w(f_B(y))$. It follows that the partition

$$\{y_1, f_A(y_1), f_B(y_1)\}, \ldots, \{y_l, f_A(y_l), f_B(y_l)\}$$

is a solution to the NMTS instance.

We remark that the size of the input of UCM is polynomial in l and w_{max}, even if all parameters are represented in unary form. Because NMTS is strongly NP-complete, UCM is also strongly NP-hard. It is easy to check that UCM under any positional scoring rule is in NP. It follows that UCM under r_{weird} is strongly NP-complete. □