Manipulating Tournaments in Cup and Round Robin Competitions*

Tyrel Russell[1] and Toby Walsh[2]

[1] Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
tcrussel@cs.uwaterloo.ca
[2] NICTA and UNSW, Sydney, Australia
toby.walsh@nicta.com.au

Abstract. In sports competitions, teams can manipulate the result by, for instance, throwing games. We show that we can decide how to manipulate round robin and cup competitions, two of the most popular types of sporting competitions in polynomial time. 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. Finally, we show that there are several different variations of standard cup competitions where manipulation remains polynomial.

1 Introduction

The Gibbard-Satterthwaite theorem proves that, under some modest assumptions, voting systems are always manipulable. One possible escape proposed by Bartholdi, Tovey and Trick is that the manipulation may be computationally too difficult to find [2] (but see [13] for discussion about whether manipulation is hard not just in the worst case). Like elections, sporting competitions can also be manipulated. For example a coalition of teams might throw games strategically to ensure that a desired team wins or a certain team loses. We consider here the computational complexity of computing such manipulations. We show that, for several common types of competitions, determining when a coalition can manipulate the result is polynomial. Our results adapt manipulation procedures for elections where voters can misrepresent their preferences. We consider two of the most common methods used for deciding sporting competitions, cups and round robins. These correspond to elections run using sequential majority voting (also known as the cup rule) and Copeland scoring, respectively.

Manipulating a sporting competition is slightly different to manipulating an election as, in a sporting competition, the voters are also the candidates. A tournament graph describes the outcome of all fair games between opponents. Manipulating a competition therefore modifies not votes but the tournament graph directly. Since it is hard without bribery or similar mechanisms for a team

* This work is funded by an NSERC Postgraduate Scholarship[1], the Department of Broadband, Communications and Digital Economy[2] and the Australian Research Council[2].

to play better than it can, we consider manipulations where teams in the coalition are only able to throw games. By comparison, in an election, voters in the manipulating coalition can mis-report their preferences in any way they choose. Tang, Shoham and Lin [11] addressed this type of tournament manipulation in team competitions by providing conditions for truthful reporting of player strengths. Their method tries to encourage teams to rank their players honestly so that, when the teams compete in bouts, the best player on one team plays the best on the other, the second best plays the opposing second and so forth. An example of this type of competition is Davis Cup Tennis.

Conitzer, Sandholm and Lang [3] give an algorithm to determine if a coalition can manipulate the cup rule. We modify this algorithm to manipulate directly the tournament graph instead of the votes. Bartholdi, Tovey and Trick [2] discuss direct manipulations of the tournament under second order Copeland, a round robin like rule with secondary tie breaking. Using the work of Kern and Paulusma [7], we show that the manipulation of round robin competitions is directly tied to the problem of winner determination in sports problems. Altman, Procaccia and Tenneholtz [1] construct a social choice rule that is monotonic, pairwise non-manipulable and non-imposing. Round robin and cup competitions are monotonic as a single team losing a game does no better. Pairwise non-manipulability means that no two teams are better off by manipulating the tournament. Our results show that round robin and cup competitions are pairwise manipulable and that manipulations can be calculated in polynomial time.

We modify our algorithms to calculate the smallest number of manipulations needed. For cup competitions, we add dynamic programming to Conitzer, Sandholm and Lang's algorithm. For round robin competitions, we modify the flow network used to solve winner determination to include weights on manipulations and calculate a minimum cost feasible flow. Vu, Altman and Shoham [12] used a similar method to calculate the probability that a team wins the competition. Vu et al. [12] provide several results on determining probabilities of teams winning given a seeding of the tournament. Hazon et al. [6] showed that it is NP-Complete to determine if a team wins a cup with a given probability. This is similar to determining a possible winner given random reseeding except edges in the tournament are labelled with probabilities. We look at the complexity of manipulation under reseeding in the deterministic case. Finally, we look at the complexity of double elimination cups.

2 Background

In many sporting competitions, the final winner of a competition is decided by a tree-like structure, called a *cup*. The most common type is a *single elimination cup*, a tree structure where the root and internal nodes represent games and leaves represent the teams in the tournament. A cup can include a *bye game*, a game where a team skips a game to re-balance the schedule. Usually, the top teams are given a bye game while the lower teams do not so that the number of teams in the next round is strict power of two. Cups need to be *seeded* to determine which teams play against each other in each round. One method for seeding is by rank. The most common method for ranked seeding or reseeding is to have the top team play the worst team, the second place team play the second worst team and so forth. An example of ranked seeding using this method is the National Basketball Association in the US. Another method for determining seeding is randomly, also known as a draw. An example of this is the UEFA Champions League where teams reaching the quarter finals are randomly paired for the remainder of the tournament. Seeding may also be more complex (for instance, it may be based on the group from which teams qualify or some other criteria). Another way that cups are modified is between fixed and unfixed cups. A *fixed cup* is a cup where there is a single seeding at the start of the cup. Examples of this are the National Basketball Association and the World Cup of Football. An *unfixed cup* is one where seeding may occur not only before the start but between any round. Examples with an unfixed cup are the National Hockey League and the UEFA Champions League.

Cups are not necessarily single elimination. A double elimination cup is designed so that a team can lose two games instead of one game. If a team loses, they play other teams that have also lost until they lose a second time or they win the final game of the tournament. These tournaments are organized as two cups where losers enter the second cup at various stages depending on when they lose their first game. Finally, a *round robin competition* is a competition where each team plays every other team a given number of times. In a *single round robin competition*, each team plays every other team exactly once. Another common variant of this is for teams to play a *double round robin competition* where each team plays every other team twice, often at home and away.

3 Manipulating the Tournament

A *tournament* is a directed graph $G = (V, E)$ where the underlying undirected graph is a complete graph. We assume that the tournament is available for the remainder of the paper. Every directed edge $(v_i, v_j) \in E$ represents a victory by v_i over v_j. The number of the teams in the competition is $|V| = m$. We define a *manipulation* of the tournament as any replacement of an edge (v_i, v_j) in the graph with the edge (v_j, v_i). This is equivalent to a manipulation of votes but here we are changing the winner directly instead of just changing the vote. Note that, as in election manipulation where the electoral vote is assumed to be known, we assume that we know, via an oracle, the relative strengths of teams and can represent the winner of the contests in the tournament graph. We restrict manipulations by only allowing the manipulation of an edge (v_i, v_j) if candidate v_i is a member of the coalition. This restricts the behaviour of the manipulators to throwing games where they could have won. This restriction is due to the fact that it is simple to perform worse but more difficult to play better. We consider two different types of manipulations. A *constructive manipulation* is one that ensures a specific team wins the competition. A *destructive manipulation* is one that ensures a specific team loses the competition. For round robin competitions, we generalize the concept of the tournament beyond the simple win-loss scoring model to a complete graph where the edge (v_i, v_j) has a non-negative weight w_{ij} which represents the number of points that would be earned by v_i when playing v_j in a fair game. We define a manipulation in this case as an outcome where the points earned in the match are different to those given by the tournament. However, manipulations are restricted so that the manipulator achieves no more points and the team being manipulated achieves no less points.

In this section, we restrict ourselves to fixed cups with a known seeding. We also look just at single round robin tournaments though the results generalize to multi-round robin tournaments.

3.1 Cup Competitions

For cup competitions, finding a constructive or destructive manipulation of the tournament is polynomial. Our results make use of results in [3] which shows that a manipulation of an election using the cup rule can be found in $O(m^3 n)$ time where m is the number of candidates and n is the number of voters.

Theorem 1. *Determining if a cup competition can be constructively manipulated using manipulations of the tournament takes polynomial time.*

Proof. This proof is a bottom up version the proof of Theorem 2 from Conitzer, Sandholm and Lang (CSL)[3] but substitutes tournament manipulations for voting manipulations. The basic CSL algorithm is a recursive method that treats each node in the tree (which is not a leaf) as a sub-election (see Algorithm CSL). Conitzer et al. [3] note that a team wins a sub-election if and only if they must win one of its children and they can defeat one of the potential winners on the other side. It is perhaps simpler to understand this algorithm from a bottom up perspective. Observe that if we have two leaf nodes v_i and v_j and there exists an arc in the tournament (v_i, v_j) then v_i wins the match and is a potential winner of the sub election between v_i and v_j. Now suppose that v_i is in the member of the coalition so it is possible for them to replace (v_i, v_j) with (v_j, v_i) in the tournament and therefore v_j is also a potential winner of the sub-election via manipulation. Assume we have some sub-election in the middle of the tournament with two sets of potential winners A and B. Any team from A is a potential winner of the sub-election if there exists a team in B that they can defeat or if a coalition member in B throws a game. The same is true for teams in B. Therefore, there is a constructive manipulation if the desired winner is a member of the potential winners at the top node in the cup tree.

The original algorithm looked at $O(m^2)$ pairs of opponents as no two teams were compared more than once. Note that the original analysis provided a looser $O(m^3)$ bound on the number of comparisons, but this can be tightened by an observation of Vu et al. [12]. The difference between direct manipulation of the tournament and the method by Conitzer, Sandholm and Lang is that determining if a team could defeat another team meant summing all values of the n voters requiring $O(n)$ time whilst in the direct manipulation of the tournament

```
Algorithm:CSL($v_w$,$c$,$T$,$C$)
input  : A team $v_w$, a cup tree $c$, a tournament graph $T$, and a coalition of
         teams $C$
output: Returns true if $v_w$ can win via manipulation and false otherwise
winners ← PossibleWinners($c$,$T$,$C$);
if $v_w \in$ winners then
    return true;
else
    return false;
```

```
Procedure:PossibleWinners($c$,$T$,$C$)
input  : A cup tree $c$, a tournament graph $T$ and a coalition of teams $C$
output: Returns the set of possible winners of the cup tree via manipulation of
        the tournament by the coalition
if leaf($c$) then
    return $\{c\}$;
else
    winners ← $\{\}$;
    LeftWinners ← PossibleWinners(left($c$) ,$T$,$C$);
    RightWinners ← PossibleWinners(right($c$) ,$T$,$C$);
    forall $v_i \in$ LeftWinners do
        if $\exists v_j \in$ RightWinners  such that  $(v_i, v_j) \in E \vee v_j \in C$ then
            add(winners,$v_i$);
    forall $v_j \in$ RightWinners do
        if $\exists v_i \in$ LeftWinners  such that  $(v_j, v_i) \in E \vee v_i \in C$ then
            add(winners,$v_j$);
    return winners;
```

this can be done in constant time. Therefore, constructive manipulation of the tournament under the cup rule takes just $O(m^2)$ time. □

We observe that destructive manipulation of a competition using tournament manipulations is similar since this simply requires determining if there is at least one other possible winner of the tournament via manipulations.

Theorem 2. *Determining if a cup tournament can be destructively manipulated using tournament manipulations takes polynomial time.*

Proof. We just determine if we can constructively manipulate the tournament for each other team in turn than the one we wish to lose. □

3.2 Round Robin Competition

For round robin competitions, manipulations of the tournament can be computed in polynomial time for a restricted class of scoring models. We define a *scoring model* to be the set of tuples giving the possible outcomes of a game. Copeland scoring has a simple win-loss ($\{(0,1),(1,0)\}$) scoring model where the wining team earns one point and the losing team earns none. Bartholdi, Tovey and Trick[2] showed that constructive manipulation can be determined in polynomial time for a chess scoring model ($\{(0,1),(\frac{1}{2},\frac{1}{2}),(1,0)\}$). Faliszewski et al. [4] showed that for a range of scoring models manipulating Copeland voting is NP-Complete.

First, we discuss the problem of determining which games need to be manipulated to ensure that a given team v_w wins the competition. Clearly, there are some games that cannot be affected by the coalition and are fixed. All other games are manipulable. Games between coalition members can earn any of the possible scores allowed by the scoring model. We restrict games against non-coalition members by only allowing the manipulator to earn less points and the non-member earns more. Determining if a given team can be made a winner is analogous to determining if a team wins a round robin tournament when the fixed games have been played and the manipulable games have not been played. The restriction of the outcomes on games between coalition and non-coalition members requires that the games have outcomes within only a subset of the scoring model. Using this observation, we obtain the following theorem.

Theorem 3. *Determining if there exists a constructive manipulation of a round robin competition is polynomial if the normalized scoring model is of the form* $S = \{(i, n-i) \mid 0 \leq i \leq n\}$ *and NP-complete, otherwise.*

Proof. This proof uses the equivalence of determining whether a team can win a tournament and determining if a constructive manipulation exists with a set of fixed and manipulable games. Note that a game between a non-coalition member v_i and a coalition member v_j is unfixed but the scores that can be assigned are restricted. When the scoring model is of the form $S = \{(i, n-i) \mid 0 \leq i \leq n\}$ and the initial result of the game is (c_i, c_j), then the remaining valid scores that can be assigned are those from (c_i, c_j) to $(n, 0)$. By normalizing this new model, we obtain one in which the non-coalition member earns c_i points by default and the result of the game is scored from the model $\{(0, c_j), \ldots, (n - c_i, 0)\}$ which is of the form $S = \{(i, n-i) \mid 0 \leq i \leq n\}$. Kern and Paulusma [7] showed that determining if a team can win a tournament (i.e. is not eliminated from competition) takes polynomial time if the normalized scoring model is of the form $S = \{(i, n-i) \mid 0 \leq i \leq n\}$ and is NP-complete otherwise. □

By comparison, it is always polynomial to determine if a destructive manipulation exists.

Theorem 4. *Determining if there is a destructive manipulation of a round robin competition takes polynomial time.*

Proof. Assume that v_l is the team that the coalition desires to lose. It is sufficient to check whether the maximum points of another team via manipulation is greater than the points of v_l. If v_l is a member of the coalition and therefore a manipulator, for each team i that we check for points, we apply only manipulations that increase the relative points between i and v_l. For all other teams, we apply the manipulation which decreases the points of v_l the most. If v_l is not a member of the coalition, no games involving v_l may be manipulated since we restrict manipulations to allow only those manipulations that increase the points of v_l and increase the relative gap between v_l and the manipulator. Therefore, no other team is better off when games involving v_l are manipulated. In both cases, we apply the manipulation that increase the points of the team under consideration against all other teams. If the total number of points of any other team is greater than the points of v_l under these manipulations, then there is a destructive manipulation of v_l. This algorithm can be run in $O(n^2)$ time. $\square$

A further complication is when the goal of manipulation is just to earn a berth in the next round of the playoffs. It is NP-hard to decide these questions under most playoff systems for all scoring models [9, 5].

4 Minimizing Manipulations

The number of manipulations required is an important factor. It may be advantageous for the coalition to manipulate as few games as possible to avoid detection or to minimize the cost of bribing players. We show that there is a polynomial algorithm to calculate manipulations which throw a minimal number of games. This highlights the vulnerability of the two most common types of competitions in sports to manipulation.

4.1 Minimal Number of Manipulations for Cup Competitions

Computing the minimal number of manipulations simply requires keeping a count within our algorithm for computing a manipulation. We give some notation to identify a specific sub-election in the cup. We let $s_\ell^{v_i}$ be the sub-election at level ℓ where v_i is a leaf node of a sub tree below $s_\ell^{v_i}$. We denote the level as the height from the bottom of the cup tree, which is assumed to be a perfect binary tree. We also define level 0 to be the level belonging to the leaves. We have m^2 constants c_{ij} that are 1 if $(v_j, v_i) \in M$ and 0 otherwise, where $M \subseteq E$ is the set of edges which can be manipulated by the coalition. This corresponds to $c_{ij} = 1$ when a manipulation must occur for v_i to win and 0 otherwise. Finally, we define the minimal number of manipulations needed to win a sub-election $s_\ell^{v_i}$, $m(v_i, s_\ell^{v_i})$, to be sum of the minimal number of the manipulations for v_i to win one of the children of $s_\ell^{v_i}$, and the minimum number of manipulations plus c_{ij} over all possible winners of the other child which v_i can defeat. We denote the set of teams that v_i can defeat either as described in the tournament or by manipulation as D_i. More formally, the minimal number of manipulations for v_i at $s_\ell^{v_i}$ ($\ell \geq 0$) is given by:

$$m(v_i, s_\ell^{v_i}) = \begin{cases} 0 & \text{if}\, \ell = 0 \\ m(v_i, s_{\ell-1}^{v_i}) + \min_{v_j \in D_i}(m(v_j, s_{\ell-1}^{v_j}) + c_{ij}) & \text{if}\, \ell > 0 \end{cases} .$$

Lemma 1. *The minimal number of manipulations needed to make a team v_i a winner at level n in the tree is equal to $m(v_i, s_n^{v_i})$.*

Proof. By induction. First, observe that the minimal number of manipulations at a leaf is 0. Hence, $m(v_i, s_0^{v_i}) = 0$ for all leaves v_i. Next note that at level 1 there are only 2 nodes in the possible winner sets of the leaves. Therefore if v_i can defeat v_j, $m(v_i, s_1^{v_i}) = m(v_i, s_0^{v_i}) + m(v_j, s_0^{v_j}) + c_{ij} = c_{ij}$ which is the exact number of manipulations that have occurred to make v_i a possible winner so far. We assume the premise for $1 < n \leq k$. Now, $m(v_i, s_{k+1}^{v_i}) = m(v_i, s_k^{v_i}) + \min_{v_j \in D_i}(m(v_j, s_k^{v_j}) + c_{ij})$. We know that $m(v_i, s_k^{v_i})$ is the minimal number of manipulations for v_i up to level k by the assumption and, for every $v_j \in D_i$, we know that $m(v_j, s_k^{v_j})$ is also the minimal number of manipulations for each v_j up to level k. By definition, c_{ij} is the number of manipulations for v_i to defeat v_j. Since v_i can defeat any v_j in D_i, the one with the fewest previous manipulations to reach k plus c_{ij} leads to the fewest manipulations in total to make v_i win the sub election $s_{k+1}^{v_i}$. This equals the minimum over the set D_i. Therefore the lemma holds for $k+1$ and, by induction, all n levels of the tree. $\square$

Theorem 5. *A modified CSL algorithm, where the team which minimizes the value of $m(v_i, s_n^{v_i})$ is selected to lose to team v_i at every node $s_n^{v_i}$, calculates the minimal number of manipulations needed to constructively or destructively manipulate a cup competition in polynomial time.*

Proof. By Lemma 1, the value of $m(v_w, s_n^{v_w})$ at the root node is the minimal number of manipulations which ensures v_w is the winner. Hence, we just need to show that the algorithm remains polynomial. The modified CSL algorithm still makes $O(m^2)$ comparisons. The only difference is that we have to calculate the minimum which can be done by storing the minimum as each team is checked. Therefore, the time complexity remains $O(m^2)$ and calculating the minimum is polynomial. Constructive manipulation requires calculating $m(v_w, s_n^{v_w})$ whilst destructive manipulation requires the minimum over all other teams. $\square$

4.2 Minimal Number of Manipulations for Round Robin Competitions

We consider here just Copeland scoring. We conjecture that similar methods could be developed for other scoring schemes.

Definition 1. *Given a tournament $T = (V, E)$ where $V = \{v_1, \ldots, v_n\}$, a set of manipulable edges $M \subseteq E$, and a distinguished node v_w, the Minimal Number of Manipulations under Copeland Scoring is the problem of determining the minimal number of edges in M that can be reversed such that $\forall_{v_k \in V, v_w \neq v_k}$ $outDegree(v_w) \geq outDegree(v_k)$.*

Note that Copeland Scoring is the simple win-loss method of scoring where the winning team earns 1 point and the losing team earns 0 points. Before we show how to calculate the minimal number of manipulations, we show that we can determine the out degree, i.e. the Copeland Score, of the distinguished node using a minimal number of manipulations in isolation with a greedy algorithm. The intuition behind this is that we select manipulations to increase the out degree of v_w.

Lemma 2. *The value of $outDegree(v_w)$ can be determined in isolation by greedily using, in sequence, a minimal number of manipulations of edges $(v_i, v_w) \in M$ where $\forall_{(v_j,v_w)\in M, v_j \neq v_i} outDegree(v_i) \geq outDegree(v_j)$ until $\forall_{v_k \in V, v_w \neq v_k}$ $outDegree(v_w) \geq outDegree(v_k)$.*

Proof. First, we prove that it always uses the least number of manipulations to increase the out degree of v_w. To reduce the out degree of two or more nodes that have an out degree larger than v_w, it takes at least two manipulations but to increase the out degree of v_w by the same amount takes just one. For a single node, it is preferred to use the manipulation involving v_w since the other node may increase the out degree of another node requiring more manipulations. Therefore, using manipulations involving v_w is most efficient.

Now we show that we never overshoot the stopping criteria and use more than a minimal number of manipulations. Assume that we use more than the minimal number of manipulations. This means that we selected an edge that did not decrease the maximum out degree when there existed an edge that would have decreased the maximum out degree of all nodes that we did not select. However, since we always selected the edge where the source node had the maximal out degree within M, we always decreased the maximum out degree whenever possible. This is a contradiction and the greedy algorithm only uses a minimal number of manipulations when reaching the stopping condition. $\square$

Theorem 6. *Determining the minimal number of tournament manipulations required under Copeland Scoring takes polynomial time.*

Proof. We define c to be the out degree of the distinguished node, v_w, calculated using the greedy algorithm. This corresponds to the number of wins earned by v_w. If the stopping condition has not been reached, we must use c to determine how many more manipulations are necessary. We construct a winner determination flow graph as described by Kern and Paulusma [7] and Gusfield and Martel[5](See Fig. 1, for example). We add a weight of 1 to each edge (v_i, v_j) where $(v_i, v_j) \notin V$ and therefore represents a manipulation. All other edges have the weight 0. The feasible flow which uses the fewest of the non-zero edges is the minimal number of tournament manipulations to achieve a constructive manipulation. Since the value of c can be determined in a linear number of steps, we only need to do a single min cost flow computation, which is polynomial, to determine the remainder of the minimum number of manipulations necessary to make v_w the team with the highest Copeland score. $\square$

Example 1. An example tournament can be seen in Fig. 1a. There are 5 teams in this tournament: v_0 to v_4. Suppose teams v_0 and v_3 form a coalition to manipulate the tournament so that v_0 wins. We want to determine the minimum

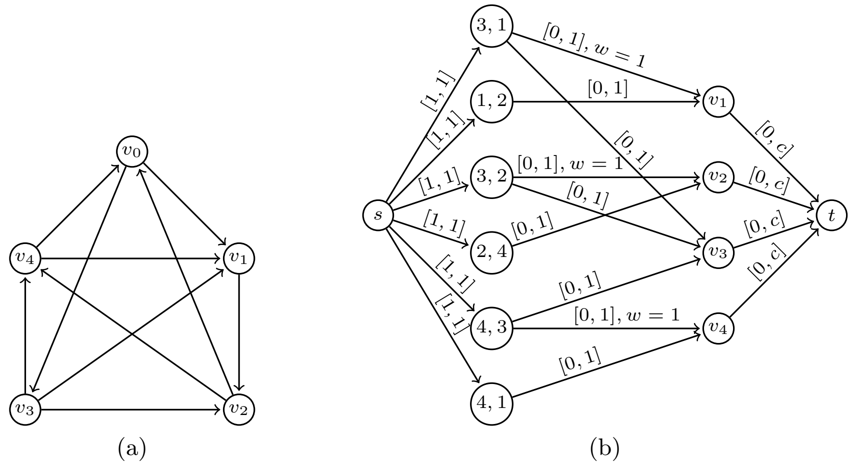

Fig. 1. (a) The tournament graph for five teams. The distinguished node in the example is v_0 which has formed a coalition with v_3. The manipulable edges are (v_3, v_1), (v_3, v_2), (v_3, v_4), (v_0, v_1) and (v_0, v_3). Edges (v_1, v_2) (v_2, v_4) and (v_4, v_1) cannot be manipulated by the coalition. (b) The min cost flow graph used to calculate the minimum number of manipulations for a given value constructed from the tournament in Fig. 1a. The distinguished team is v_0, $c = 2$ and all weights not shown are 0.

number of manipulations needed to ensure that v_0 is the winner. This requires switching any of the arcs where team v_3 wins. We know that the value of $c = 2$ since none of v_0's edges are manipulable in v_0's favour. We construct the graph seen in Fig. 1b to determine for $c = 2$ if there is a feasible solution. The solution returned has a minimum cost which is equal to the minimum number of manipulations needed to get a feasible flow with the value c plus any used in the greedy algorithm.

5 Reseeding

If we add multiple seeding rounds then computing a manipulation appears difficult. Recall that ranked reseeding matches the best remaining teams against the worst remaining teams in each round. The CSL algorithm cannot therefore be applied and a general solution is not known. However, if the size of the coalition is a constant c, then we can determine a manipulation in polynomial time.

Theorem 7. *For a ranked reseeding cup competition, if the manipulating coalition is of bounded size c, then determining a set of manipulations that makes a team win takes polynomial time.*

Proof. The key observation is that with a constant sized coalition there are only a polynomial number of ways to manipulate the games by rearranging the tournament graph. It suffices to check the winner of each of the polynomial number of fixed tournament graphs. For each fixed tournament graph, the winner can be determined in linear time as there are only $O(m)$ matches to check.

We show that there are only a polynomial number of different arrangements of manipulations. First note that at most c of the $\frac{m}{2}$ matches in the first round have more than one team as a possible winner. This means that there is at most 2^c possibilities to examine after each round. As there are $\log(m)$ rounds, we consider at most $(2^c)^{\log m}$ ($=m^c$) possibilities. Hence there are at most $O(m^c)$ arrangements of manipulations for an unfixed cup with ranked reseeding and a constant sized coalition. It is sufficient to check each arrangement, which can be done in linear time. This gives a polynomial algorithm for bounded c. □

With random reseeding the problem can be separated into two issues: determining whether manipulation is possible to make a team a winner under every possible seeding and determining if there exists any seeding such that the coalition can manipulate the games to make a given team the winner. It is unknown whether either of these problems have polynomial algorithms. Vu et al. [12] and Hazon et al. [6] tackle some probabilistic variants of possible winners without manipulation of games. However, the complexity of determining possible winners with a win-loss tournament graph in balanced cup trees remains open [8, 6, 10].

6 Double Elimination Competitions

In a double elimination competitions, a manipulation of the tournament does not automatically bounce the manipulator out of the tournament as in the single elimination case. However, it does guarantee that the manipulator will be bounced to the secondary bracket from the primary bracket on the first manipulation and out of the tournament on the second manipulation. As in the case of ranked reseeding, a general solution is not known but there is a polynomial algorithm for double elimination tournaments if the coalition is of constant size.

Theorem 8. *For double elimination tournaments, if the coalition is of a constant size c, determining whether there is a constructive manipulation takes polynomial time.*

Proof. This proof follows similar lines as the proof for ranked reseeding. We will show that there is a polynomial number of manipulation scenarios which can be checked in linear time. If there is a coalition of size c then a team can manipulate the cup only once if they wish to win the tournament and twice if they desire another team to win. At each step in the tree, a team must decided whether they wish to manipulate or not. Before and after they have manipulated once, there remains c teams which can manipulate. Only after they have manipulated a second time are they removed from the competition. This means there are at most 2^c manipulations at each of the $logm$ levels. This gives us $O(2^{\log mc})(= O(m^c))$ possibilities that can be checked in linear time, which gives a polynomial algorithm for determining if there is a constructive manipulation. □

7 Conclusions and Open Problems

In sporting tournaments, teams can directly manipulate the tournament graph. We showed that algorithms used to compute manipulations of votes in elections can be modified to determine the manipulations needed of the tournament graph. We proved that such direct manipulation of the fixed cup and round robin competitions can be computed in polynomial time. In a similar way, we can determine the minimal number of manipulations needed. For ranked reseeding of cup competitions, we showed that it is easy to calculate the number of manipulations if the size of the manipulating coalition is bounded by a constant. We also gave a polynomial time algorithm for double elimination tournaments for a constant sized coalition. A number of open question remain. The manipulation of various variations of the cup competition have unknown complexity including the ranked and random cup competitions. For random cup competitions, the complexity of manipulation is also unknown if the size of the coalition is bounded. Similarly, the complexity of manipulating double elimination competitions is still undetermined when the size of the coalition is unbounded.

References

1. A. Altman, A. D. Procaccia, and M. Tennenholtz. Nonmanipulable Selections from a Tournament. In *Proc. of 21st Intl. Joint Conf. on AI* 2009.
2. J. J. Bartholdi, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.
3. V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *JACM*, 54:1–33, 2007.
4. P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: Ties matter. In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008.
5. D. Gusfield and C. E. Martel. The structure and complexity of sports elimination numbers. *Algorithmica*, 32:73–86, 2002.
6. N. Hazon, P. E. Dunne, S. Kraus, and M. Wooldridge. How to Rig Elections and Competitions. In *Proc. of 2nd Int. Workshop on Computational Social Choice*, 2008.
7. W. Kern and D. Paulusma. The computational complexity of the elimination problem in generalized sports competitions. *Discrete Optimization*, 1:205–214, 2004.
8. J. Lang, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Winner Determination in Sequential Majority Voting. In *Proc. of the 20th Int. Joint Conf. on AI*, 2007.
9. S. T. McCormick. Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Operations Research*, 47:744–756, 1999.
10. M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Dealing with incomplete agents' preferences and an uncertain agenda in group decision making via sequential majority voting. In *Proc. of the 11th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 2008.
11. P. Tang, Y. Shoham, and F. Lin. Team Competition. In *Proc. of 8th Intl. Conf. on Autonomous Agents and Multiagent Systems*, 2009.
12. T. Vu, A. Altman, and Y. Shoham. On the Complexity of Schedule Control Problems for Knockout Tournaments. In *Proc. of 8th Intl. Conf. on Autonomous Agents and Multiagent Systems*, 2009.
13. T. Walsh Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In *Proc. of 21st Intl. Joint Conf. on AI* 2009.