Chapter 3 Bribery and Manipulation with Uncertain Information

This chapter details work on probabilistic models of lobbying in environments with multiple referenda and sports tournaments. This work includes model building, aggregation methods, and complexity results for the given models and methods.

Section 3.1 develops a novel and flexible model to represent majority voting with uncertain information. In this section we provide a classification of the complexity of finding efficient bribery schemes given a set of voters. Their individual preferences are represented as probability distributions over a set of issues, their prices for changing their preferences, and a budget. Much of this material is available both as a conference publication [45] and a longer version is available as a technical report [46]. This first section also develops three different ways in which an outside actor can channel money to the individual voters and establishes three different criteria to evaluate the outcome of a vote in settings with uncertain information. We show that, depending on the particular combination of evaluation and bribery models chosen, the nine problems range in complexity from polynomial time to NP-complete; these results are summarized in Table 3.1 and Table 3.2. This difference reveals that modeling choices can have significant effects on the complexity of calculating efficient bribery schemes.

Section 3.2 develops a novel and flexible model to represent sports tournaments and competitions. In this section we show a classification of the complexity of finding efficient bribery schemes for three common types of sports tournaments over five distinct problem variants given a set of teams, their probabilities of each possible win, and prices for decreasing their competitive output (purposefully losing or underperforming in a match). This work has been previously published and is available as a conference publication [87]. The evaluation complexity of these problems range from polynomial time to NP^{PP} and is summarized in Table 3.3. The results show that in some cases the added uncertainty increases the complexity of manipulating sports tournaments, while in other cases it does not. While this increase in complexity is not uniform across all tournament types, the change shows strong evidence that reasoning in domains with uncertain data leads to an increase in reasoning complexity.

3.1 Majority Voting and Multiple Referenda

In most democratic political systems, laws are passed by elected officials who are supposed to represent their constituencies. Individual entities such as citizens or corporations are not supposed to have undue influence in the wording or passage of a law. However, they are allowed to make contributions to representatives, and it is common to include an indication that the contribution carries an expectation that the representative will vote a certain way on a particular issue.

Many factors can affect a representative's vote on a particular issue. There are the representative's personal beliefs about the issue, which presumably were part of the reason that the constituency elected them. There are also the campaign contributions, communications from constituents, communications from potential donors, and the representative's own expectations of further contributions and political support [83].

It is a complicated process to reason about. Earlier work (see the references given in Chapter 2) considered the problem of meting out contributions to representatives in order to pass a set of laws or influence a set of votes. However, the earlier computational complexity work by Christian et al. [24] and others [52, 110] on this problem made the assumption that a politician who accepts a contribution will in fact—if the contribution meets a given threshold—vote according to the wishes of the donor.

It is said that "an honest politician is one who stays bought," but that does not take into account the ongoing pressures from personal convictions and opposing lobbyists and donors. We consider the problem of influencing a set of votes under the assumption that we can influence only the *probability* that the politician votes as we desire.

There are several axes along which the picture is complicated in realistic scenarios. To describe these formally, we introduce various evaluation criteria and bribery methods. The first is the notion of sufficiency: What does it mean to say we have donated enough to influence the vote? Does it mean that the probability that a single vote will go our way is greater than some threshold, or that the probability that all the votes go our way is greater than that threshold? We formally define and discuss these and other criteria in the section on evaluation criteria (Section 3.1.3). In particular, we consider three methods for evaluating the outcome of a vote given voters' probability of voting "yes" on a particular issue.

Strict Majority (SM): A vote on an issue is won by a strict majority of voters having a probability of accepting this issue that exceeds a given threshold.

Average Majority (AM): A vote on an issue is won exactly when the voters' average probability of accepting this issue exceeds a given threshold.

Probabilistic Majority (PM): A vote on an issue is won exactly when the sum of the probabilities of possible futures (i.e., of possible scenarios in which a strict majority of voters accepts this issue) exceeds a given threshold.

How does one donate money to a campaign? In the United States there are several laws that influence how, when, and how much a particular person or organization can donate to a particular candidate. We examine ways in which money can be channeled into the political process in the section on bribery methods (Section 3.1.2). In particular, we consider three methods that an actor (called "The Lobby") can use to influence the voters' preferences of voting for or against multiple issues.

Microbribery (MB): The Lobby may choose which voter to bribe and on which issue in order to influence the outcome of the vote.

Issue Bribery (IB): The Lobby may choose which issues to support, and, for each issue supported, the funds are equally distributed over all the voters.

Voter Bribery (VB): The Lobby may choose which voters to bribe, and, for each voter bribed, the funds are equally distributed over all the issues.

The voter bribery method is due to Christian et al. [24], who were the first to study lobbying in the context of direct democracy where voters vote on multiple referenda. Their "Optimal Lobbying" problem (denoted OL) is a deterministic and unweighted variant of the lobbying problems that we present in this chapter. A generalized problem described by Christian et al., the "Optimal Weighted Lobbying" (OWL) problem, which allows different voters to have different prices and so generalizes OL, can be expressed as and solved via the "binary multi-unit combinatorial reverse auction winner-determination problem"(see [114] for its definition).

The microbribery method in the context of lobbying—though inspired by the different notion of microbribery than Faliszewski et al. [53–55] introduced in the context of bribery in voting—is new to this paper. As described in this section, microbribery more closely resembles the bribery methods discussed in Faliszewski [51] and Christian et al. [24] than any other existing model as we allow for individually priced voters and the novel addition of allowing voters prices to range based on the amount of change.

In Section 3.1.1 we formally describe our model of reasoning about bribery in voting with multiple referenda where voters express their probabilities of voting for or against each issue. In Section 3.1.2 we formally describe the three bribery methods at the disposal of The Lobby. In Section 3.1.3 we then describe three different evaluation criteria for this scenario. Section 3.1.4 formally states the nine decision problems we study in this domain while Section 3.1.5 formally states an instance of the problem where The Lobby expresses their preferences as weights over individual issues. Section 3.1.6 details our complexity results and ends with a few observations about the change in reasoning complexity when models move from deterministic setting to models of uncertain information.

3.1.1 Initial Model

We begin with a simplistic version of the PROBABILISTIC LOBBYING PROBLEM (PLP, for short), in which voters start with initial probabilities of voting for an issue and are assigned known costs for increasing their probabilities of voting according to The Lobby's[1] agenda by each of a finite set of increments. The question, for this class of problems, is: given the above information, along with an agenda and a fixed budget B, can The Lobby target its bribes in order to achieve its agenda?

The complexity of the problem seems to hinge on the evaluation criterion for what it means to "win a vote" or "achieve an agenda." We discuss the possible interpretations of evaluation and bribery later in this section. First, however, we will formalize the problem by defining data objects needed to represent the problem instances. A similar model was first discussed by Reinganum [110] in the continuous case and we translate it here to the discrete case. This will allow us to present algorithms for, and a complexity analysis of, the problem.

Let $\mathbb{Q}_{[0,1]}^{m\times n}$ denote the set of $m\times n$ matrices over $\mathbb{Q}_{[0,1]}$ (the rational numbers in the interval $[0,1]$). We say $P \in \mathbb{Q}_{[0,1]}^{m\times n}$ is a probability matrix (of size $m\times n$), where each entry $p_{i,j}$ of P gives the probability that voter v_i will vote "yes" for referendum (synonymously, for issue) r_j. The result of a vote can be either a "yes" (represented by 1) or a "no" (represented by 0). Thus, we can represent the result of any vote on all issues as a $0/1$ vector $\vec{X} = (x_1, x_2, \ldots, x_n)$, which is sometimes also denoted as a string in $\{0,1\}^n$.

We associate with each voter/issue pair (v_i, r_j) a discrete price function $c_{i,j}$ for changing v_i's probability of voting "yes" for issue r_j. Intuitively, $c_{i,j}$ gives the cost for The Lobby of raising or lowering (in discrete steps) the ith voter's probability of voting "yes" on the jth issue. A formal description is as follows.

[1] In this chapter we use The Lobby to represent any outside agent attempting to manipulate the vote. This is in keeping with other work in the ComSoc community [52].

Given the entries $p_{i,j} = {}^{a_{i,j}}/_{b_{i,j}}$ of a probability matrix $P \in \mathbb{Q}^{m\times n}_{[0,1]}$, where $a_{i,j} \in \mathbb{N} = \{0,1,\ldots\}$, $b_{i,j} \in \mathbb{N}_{+} = \{1,2,\ldots\}$, and $a_{i,j} \leq b_{i,j}$, choose some $k \in \mathbb{N}$ such that $k+1$ is a common multiple of all $b_{i,j}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, and partition the probability interval $[0,1]$ into $k+1$ steps of size ${}^{1}/_{(k+1)}$ each.[2] The integer k will be called the discretization level of the problem. For each $i \in \{1,2,\ldots,m\}$ and $j \in \{1,2,\ldots,n\}$, $c_{i,j} : \{0, {}^{1}/_{(k+1)}, {}^{2}/_{(k+1)}, \ldots, {}^{k}/_{(k+1)}, 1\} \rightarrow \mathbb{N}$ is the *(discrete) price function for* $p_{i,j}$, i.e., $c_{i,j}({}^{\ell}/_{(k+1)})$ is the price for changing the probability of the ith voter voting "yes" on the jth issue from $p_{i,j}$ to ${}^{\ell}/_{(k+1)}$, where $0 \leq \ell \leq k+1$. Note that the domain of $c_{i,j}$ consists of $k+2$ elements of $\mathbb{Q}_{[0,1]}$ including 0, $p_{i,j}$, and 1. In particular, we require $c_{i,j}(p_{i,j}) = 0$, i.e., a cost of zero is associated with leaving the initial probability of voter v_i voting on issue r_j unchanged. Note that $k = 0$ means $p_{i,j} \in \{0,1\}$, i.e., in this case each voter either accepts or rejects each issue with certainty and The Lobby can only flip these results. This special case in our problem definition encompasses the Optimal Lobbying problem of Christian et al. [24]. The image of $c_{i,j}$ consists of $k+2$ nonnegative integers including 0, and we require that, for any two elements a, b in the domain of $c_{i,j}$, if $p_{i,j} \leq a \leq b$ or $p_{i,j} \geq a \geq b$, then $c_{i,j}(a) \leq c_{i,j}(b)$. This guarantees monotonicity on the prices.

We represent the list of price functions associated with a probability matrix P as a table C_P, called the cost matrix, whose $m \cdot n$ rows give the price functions $c_{i,j}$ and whose $k+2$ columns give the costs $c_{i,j}({}^{\ell}/_{(k+1)})$, where $0 \leq \ell \leq k+1$. Note that we choose the same k for each $c_{i,j}$, so we have the same number of columns in each row of C_P. The entries of C_P can be thought of as "price tags" indicating what The Lobby must pay in order to change the probabilities of voting.

The Lobby also has an integer-valued budget B and an "agenda," which we will denote as a vector $\vec{Z} \in \{0,1\}^n$ for n issues, containing the outcomes The Lobby would like to see on these issues. For The Lobby, the prices for a bribery that moves the outcomes of a

[2] There is some arbitrariness in this choice of k. One might think of more flexible ways of partitioning $[0,1]$. We have chosen this way for the sake of simplifying the representation, but we mention that all that matters is that for each i and j, the discrete price function $c_{i,j}$ is defined on the value $p_{i,j}$, and is set to zero for this value.

referendum in the wrong direction do not matter. Hence, if $\vec{Z}$ is zero at position j, then we can set $c_{i,j}(a) = --$ (indicating an unimportant entry) for $a > p_{i,j}$, and if $\vec{Z}$ is one at position j, then we can set $c_{i,j}(a) = --$ (indicating an unimportant entry) for $a < p_{i,j}$. Without loss of generality, we can also assume that $c_{i,j}(a) = 0$ if and only if $a = p_{i,j}$.

For simplicity, we may assume that The Lobby's agenda is all "yes" votes, so the target vector is $\vec{Z} = 1^n$. This assumption can be made without loss of generality, since if there is a zero in $\vec{Z}$ at position j, we can flip this zero to one and also change the corresponding probabilities $p_{1,j}, p_{2,j}, \ldots, p_{m,j}$ in the jth column of P to $1 - p_{1,j}, 1 - p_{2,j}, \ldots, 1 - p_{m,j}$. (See the evaluation criteria in Section 3.1.3 for how to determine the result of voting on a referendum.) Moreover, the rows of the cost matrix C_P that correspond to issue j have to be mirrored, that is, the prices have been flipped so we are attempting to achieve all yes' instead of a mix of yes and no votes.

Example 3.1.1 *Consider the following problem instance with $k = 9$ (so there are $k+1 = 10$ steps), $m = 2$ voters, and $n = 3$ issues. We will use this as a running example for the rest of this section. In addition to the above definitions for k, m, and n, we give the following probability matrix P and cost matrix C_P for P. This example is normalized for an agenda of $\vec{Z} = 1^3$, which is why The Lobby has no incentive for lowering the acceptance probabilities, so those costs are omitted below.*

Our example consists of a probability matrix P:

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4

and the corresponding cost matrix C_P*:*

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

In Section 3.1.2, we describe three bribery methods which are three specific ways in which The Lobby can influence the voters. These will be referred to as *microbribery* (MB), *issue bribery* (IB), and *voter bribery* (VB). In addition to the three bribery methods described in Section 3.1.2, we define three ways of evaluating a set of votes. These evaluation criteria are defined in Section 3.1.3 and will be referred to as *strict majority* (SM), *average majority* (AM), and *probabilistic majority* (PM). It is important to formalize the notion of winning in this problem due to our modeling of uncertainty. Given different types of information agents can choose to optimize over different realizations of systems that contain uncertainty. This method of examining different decision strategies is in keeping with other works on game theory and decision making under uncertainty, see Luce and Raiffa for a more complete treatment [84]. The nine basic probabilistic lobbying problems we will study (each a combination of MB/IB/VB bribery under SM/AM/PM evaluation) are defined in Section 3.1.4, and a modification of these basic problems with additional issue weighting is introduced in Section 3.1.5.

3.1.2 Bribery Methods

We begin by formalizing the bribery methods by which The Lobby can influence votes on issues. We will define three methods for donating this money.

Microbribery (MB)

The first method at the disposal of The Lobby is *microbribery*. Though our notion of microbribery was inspired by the work of Faliszewski et al. [53–55], it should not be confused with their definition of the term "microbribery," used in the context of bribing "irrational" voters in Llull/Copeland elections. In the Llull/Copeland elections, voters are represented via binary preference relations that may or may not be transitive. Microbribery in the model defined by Faliszewski et al. [53–55] allows the briber to flip single entries in the voters' preference tables possibly making each voter irrational (a voter with non-transitive preferences). Microbribery is the editing of individual elements of the P matrix according to the costs in the C_P matrix. Thus The Lobby picks both which voter to influence and on which issue to influence that voter. This bribery method allows the most flexible version of bribery defined in this document. It generally models private donations made to candidates in support of specific issues from either Political Action Committees (PACs) or interested individual parties. This method of contribution, in some cases, has an impact on an individual's platform or voting beliefs [48].

More formally, if voter i is bribed with d dollars on issue j, then all entries $c_{i,j}[\ell]$ are updated as follows:

$$c_{i,j}[\ell] := \begin{cases} -- & \text{if}(c_{i,j}[\ell] = --) \vee ((c_{i,j}[\ell] - d) \leq 0) \\ c_{i,j}[\ell] - d & \text{if}(c_{i,j}[\ell] - d) > 0. \end{cases}$$

We also need to update P with the correct discrete price step. To do this we determine the maximum "$--$" entry in $c_{i,j}$, $U = \max\{x | c_{i,j}[x] = --\}$. We then update $c_{i,j}[x] = 0$ and set $p_{i,j} = {}^{x}/_{(k+1)}$.

Example 3.1.2 *Take our running example. In order to see the effect of MB, imagine The Lobby were to donate* $\$100$ *to voter* v_1 *on issue* r_2*. This would raise the probability of* v_1 *voting yes on* r_2 *to* 0.6*. The updated P matrix* P' *and updated* C_P *matrix* C'_P *are:*

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.6	0.5
v_2	0.4	0.7	0.4

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	−−	−−	−−	0	40	210	420	500
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Issue Bribery (IB)

The second method at the disposal of The Lobby is *issue bribery*. We can see from the P matrix that each column represents how all voters think about a particular issue. In this method of bribery, The Lobby can pick a column of the matrix and edit it according to some budget. The money will be equally distributed among all the voters and the voter probabilities will move accordingly. So, for d dollars donated, each voter receives a fraction of d/m and his or her probability of voting "yes" changes accordingly. This can be thought of as special-interest group donations. Special-interest groups such as PETA[3] focus on issues and dispense their funds across an issue rather than by voter. The bribery could be funneled through such groups.

This method of influence is demonstrated in the US political system through the use of Super PACs in light of the US Supreme Court ruling, *Citizen United v. Federal Election Commission*. This controversial decision allows for almost unlimited campaign contribu-

[3] People for the Ethical Treatment of Animals, a narrow-focus group that protests animal testing of food and drugs, and the swatting of flies.

tions from interested parties to Super PACs; which are political action committees designed to generally support a narrow set of issues through the use of direct lobbying efforts and media campaigns.

We require the elements of C_P to be discrete, so we must place some restrictions on this method of bribery in order to avoid fractional dollars. Specifically, we assume that bribery will be donated in multiples of m, the number of candidates. In this way only integer numbers of dollars will be donated per voter. Example 3.1.3 illustrates the process.

Example 3.1.3 *Take our running example. In order to see the effect of IB, imagine The Lobby were to donate* \$140 *to issue* r_1. *Since there are two voters this means that* \$70 *is donated to each of* v_1 *and* v_2 *on issue* r_1. *The updated* P *matrix* P' *and updated* C_P *matrix* C'_P *are:*

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	0.7	0.4

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	30	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	−−	−−	−−	0	50	130	200
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Voter Bribery (VB)

The third method at the disposal of The Lobby is *voter bribery*. In the P matrix, each row represents how an individual voter will cast his ballot for all issues on the docket. In this method of bribery, The Lobby picks a voter and then pays to edit the entire row at once with the funds being equally distributed over all the issues. So, for d dollars a fraction of ${}^{d}/_{n}$ is spent on each issue, which moves accordingly. The cost of moving the voter is given by the C_P matrix as before. This method of bribery is analogous to “buying” or pushing a single politician or voter. The Lobby seeks to donate so much money to some individual voters that they have no choice but to move all of their votes toward The Lobby’s agenda.

This method of general lobbying occurs in the US political system [48], though not always in the form of monetary exchange. As discussed by Hall and Wayman [69], oftentimes an interested party will provide education or information about a broad set of issues to a voter. This information, in many cases, is biased in support of the interested parties position on the legislation. This method of intervention can change a voter’s opinion across a large set of issues.

Again, we require the elements of C_P to be discrete so we must place some restrictions on this method of bribery in order to avoid fractional dollars. Specifically, we assume that bribery will be donated in multiples of n, the number of issues. In this way only integer numbers of dollars will be donated per voter. Example 3.1.4 illustrates the process.

Example 3.1.4 *Take our running example. In order to see the effect of VB, imagine The Lobby were to donate* \$270 *to voter* v_2*. Since there are three issues this means that* \$90 *is donated to each of the three issues for* v_2*. The updated* P *matrix* P' *and updated* C_P *matrix* C_P' *are:*

$$P' = \begin{array}{|c||c|c|c|}
\hline
 & r_1 & r_2 & r_3 \\
\hline
v_1 & 0.8 & 0.3 & 0.5 \\
\hline
v_2 & 0.7 & 1.0 & 0.6 \\
\hline
\end{array}$$

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	−−	−−	−−	−−	30	110	160
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	−−	−−	−−	0
$c_{2,3}$	−−	−−	−−	−−	−−	−−	0	10	90	210	360

Observe that microbribery is equivalent to issue bribery if there is only one voter. Similarly, microbribery is equivalent to voter bribery if there is only one referendum.

3.1.3 Evaluation Criteria

Defining criteria for how an issue is won is the next important step in formalizing our models. Here we define three methods that one could use to evaluate the eventual outcome of a vote. Since we are focusing on problems that are probabilistic in nature, it is important to note that no evaluation criterion will guarantee a win. The criteria below yield different outcomes depending on the model and problem instance.

Strict Majority (SM)

For each issue, a strict majority of the individual voters have probability greater than some threshold, t, of voting according to the agenda.

In Example 3.1.1, with $t = 0.5$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
SM	0	0	0

None of the issues has a strict majority of voters with above a 0.5 probability of voting "yes" in this setting and thus, The Lobby has not achieved its agenda. However, if we look at the result for Example 3.1.4, with $t = 0.5$ after the illustrated round of VB we have:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
SM	1	0	0

While The Lobby has still not achieved its agenda, it has moved closer to its desired result with the selected bribery action since there is now one issue, r_1, which has a strict majority of voters with $P_{i,1} > 0.5$ probability of voting in accordance with The Lobby.

Average Majority (AM)

For each issue r_j of a given probability matrix P, we define the average probability $\overline{p_j} = (\Sigma_{i=1}^{m} p_{i,j})/m$ of voting "yes" for r_j. We can now evaluate the vote to say that r_j is accepted if and only if $\overline{p_j} > t$ where t is some threshold.

In Example 3.1.1, with $t = 0.5$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
$\overline{p_j}$	0.6	0.5	0.45
AM	1	0	0

This table is augmented with the resultant probability ($\overline{p_j}$) and the AM result for $t = 0.5$. The Lobby has achieved exactly one of its desired results in this example. However, if we investigate the voting result from Example 3.1.4 with $t = 0.5$ we have the following result after a round of VB:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
$\overline{p_j}$	0.75	0.65	0.55
AM	1	1	1

In this example the round of VB has been successful, The Lobby has fully achieved its agenda through bribery.

Probabilistic Majority (PM)

The third criterion takes into account the probabilities of possible *scenarios*, or possible futures. Each possible scenario, in which voters commit to "yes" or "no" votes for each issue, has a probability. Under this criterion, the probability that The Lobby's agenda passes is the sum of probabilities of those scenarios in which the agenda passes. We call these majority scenarios and denote their issue-wise probability with $\vec{S} \in \mathbb{Q}_{[0,1]}$.

More formally, for each issue r_j of a given probability matrix P, we want to find the sum of the probabilities of those futures in which a strict majority of voters vote "yes" for r_j. We say that there is a probabilistic majority for r_j if the probability of receiving a majority of "yes" votes exceeds a threshold t. A similar possible futures evaluation metric is used by [71].

In our running example there are only two voters and, therefore, only one scenario, for each issue, in which a strict majority is reached. This is the situation where both voters cast "yes" ballots for both issues. We can compute these values by multiplying the probabilities of "yes" votes by each other ($0.8 \cdot 0.4$ for r_1 etc.). In Example 3.1.1, with $t = 0.2$ we would have the following result:

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4
S_i	0.32	0.21	0.20
PM	1	1	0

This table is augmented with the resultant probability of a majority scenario (S_i) and the PM result for $t > 0.2$. The Lobby has achieved its desired effect on r_1 and r_2. However, if we investigate the voting result from Example 3.1.4 with $t = 0.5$ we have the following result after a round of VB:

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.7	1.0	0.6
S_i	0.56	0.30	0.30
PM	1	1	1

In this example the round of VB has been successful, The Lobby has fully achieved its agenda through bribery.

The examples shown here contain only two voters and, therefore, only one majority scenario for each issue. When there are more than two voters the computation of the probability of a majority scenario is not so straightforward.

Observe that all three evaluation criteria coincide if there is only one voter or if the discretization level equals zero, so the problem is deterministic.

3.1.4 Basic Probabilistic Lobbying Problem

We can now introduce the nine basic problems that we will study. For $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$, we define the following problem.

Name: X-Y PROBABILISTIC LOBBYING PROBLEM.

Given: A probability matrix $P \in \mathbb{Q}_{[0,1]}^{m \times n}$ with a cost matrix C_P (with integer entries), a budget B, and some threshold $t \in \mathbb{Q}_{[0,1]}$.

Question: Is there a way for The Lobby to influence P using bribery method X and evaluation criterion Y, without exceeding budget B, such that the result of the votes on all issues equals 1^n?

We abbreviate this problem name as X-Y-PLP. Observe that the discretization level is an implicit, unary parameter of the problem. It is indirectly specified through the given cost matrix C_P for a problem instance.

Example 3.1.5 *Recall our running Example 3.1.1. We have the following matrices for P and C_P:*

$P =$

	r_1	r_2	r_3
v_1	0.8	0.3	0.5
v_2	0.4	0.7	0.4

$C_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

If we have microbribery with the average majority criterion and $t = 0.5$ with the matrices considered above, then The Lobby needs to select some voters to bribe in order to achieve its agenda. If the budget is set for \$50*, then we have an instance of* MB-AM-PLP *with* $(P, C_P, 50, 0.5)$ *with P and C_P as shown above and target vector $\vec{Z} = 1^n$. The Lobby* *needs to bribe v_1 on r_2 with a payment of* \$10 *and v_1 on r_3 with a payment of* \$25. *The updated matrices are:*

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	−−	0	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	−−	−−	0	45	65	125
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

$P' =$

	r_1	r_2	r_3
v_1	0.8	0.4	0.7
v_2	0.4	0.7	0.4
$\overline{p_j}$	0.6	0.55	0.55
AM	1	1	1

Each referendum passes according to the AM evaluation criteria and therefore $(P, C_P, 50, 0.5)$ *is in* MB-AM-PLP.

3.1.5 Issue Weighting

We augment the model to include the concept of issue weighting. It is reasonable that certain issues will be of more importance than others. For this reason we will allow The Lobby to assign higher weights to the issues that they deem more important. The lobby no longer states an agenda, rather they have weights over the issues and a target total weight. These positive integer weights will be defined for each issue.

We will specify these weights as a vector $\vec{W} \in \mathbb{N}^n_+$ of length n, the total number of issues in our problem instance. The higher the weight, the more important that particular issue is to The Lobby. Along with the weights for the issues we are also given an objective value $V \in \mathbb{N}_+$, which is the minimum weight The Lobby wants to see passed. We allow this set to be a partial order (a reflexive, transitive, and antisymmetric ordering) over the weights. Therefore, it is possible for The Lobby to have an ordering such as $w_1 = w_2 = \cdots = w_n$. If this is the case, and $V = n$, we are left with an instance of X-Y-PLP, where $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$.

We now introduce the nine probabilistic lobbying problems with issue weighting. For $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$, we define the following problem.

Name: X-Y PROBABILISTIC LOBBYING PROBLEM WITH ISSUE WEIGHTING.

Given: A probability matrix $P \in \mathbb{Q}^{m \times n}_{[0,1]}$ with cost matrix C_P an issue weight vector $\vec{W} \in \mathbb{N}^n_+$, an objective value $V \in \mathbb{N}_+$, and a budget B.

Question: Is there a way for The Lobby to influence P using bribery method X and evaluation criterion Y, without exceeding budget B such that the total weight of all issues that pass is at least V?

We abbreviate this problem name as X-Y-PLP-WIW.

Example 3.1.6 *Consider our running example, now augmented with* $\vec{W} = \langle 5, 10, 10 \rangle$. *We now provide an augmented P matrix which includes our vector of weights.*

$$P = \begin{array}{|c||c|c|c|}
\hline
 & r_1 & r_2 & r_3 \\
\hline
w_i & 5 & 10 & 10 \\
\hline
\hline
v_1 & 0.8 & 0.3 & 0.5 \\
\hline
v_2 & 0.4 & 0.7 & 0.4 \\
\hline
\end{array}$$

$C_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	100	140
$c_{1,2}$	−−	−−	−−	0	10	70	100	140	310	520	600
$c_{1,3}$	−−	−−	−−	−−	−−	0	15	25	70	90	150
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

If we have microbribery with the average majority criterion and $t = 0.5$ with the matrices considered above, then The Lobby needs to select some voters to bribe in order to achieve its objective value $V = 25$. If the budget is set for $75, *then we have an instance of* VB-AM-PLP-WIW *with* $(P, C_P, 75, 0.5, 25)$ *with P and C_P as shown above. The Lobby needs to bribe v_1 with a payment of* $75. *This payment will be split evenly over all the issues for v_1. The updated matrices are:*

$C'_P =$

$c_{i,j}$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$c_{1,1}$	−−	−−	−−	−−	−−	−−	−−	−−	0	75	115
$c_{1,2}$	−−	−−	−−	−−	0	45	75	115	285	495	575
$c_{1,3}$	−−	−−	−−	−−	−−	−−	−−	0	45	65	125
$c_{2,1}$	−−	−−	−−	−−	0	30	40	70	120	200	270
$c_{2,2}$	−−	−−	−−	−−	−−	−−	−−	0	10	40	90
$c_{2,3}$	−−	−−	−−	−−	0	70	90	100	180	300	450

Table 3.1: Complexity results for the X-Y PROBABILISTIC LOBBYING PROBLEM, where $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$

Problem	Classical Complexity	Theorem or Corollary
MB-SM-PLP	P	Thm. 3.1.8
MB-AM-PLP	P	Thm. 3.1.9
MB-PM-PLP	$\in$ NP	Thm. 3.1.10 and 3.1.11
IB-SM-PLP	P	Thm. 3.1.12
IB-AM-PLP	P	Thm. 3.1.12
IB-PM-PLP	$\in$ NP	Thm. 3.1.10
VB-SM-PLP	NP-complete	Thm. 3.1.13
VB-AM-PLP	NP-complete	Thm. 3.1.13
VB-PM-PLP	NP-complete	Thm. 3.1.14

$$P' = \begin{array}{|c||c|c|c|}
\hline
 & r_1 & r_2 & r_3 \\
\hline
w_i & 5 & 10 & 10 \\
\hline
\hline
v_1 & 0.8 & 0.4 & 0.7 \\
\hline
v_2 & 0.4 & 0.7 & 0.4 \\
\hline
\hline
\overline{p_j} & 0.6 & 0.55 & 0.55 \\
\hline
AM & 1 & 1 & 1 \\
\hline
\end{array}$$

In this case all the referenda have passed and so The Lobby has achieved its objective value $V = 25$. Therefore, $(P, C_P, 50, 0.5, 25)$ is in VB-AM-PLP-WIW.

3.1.6 Results

In this section we report some results from G. Erdélyi et al. [45]. There are additional results from this paper which we do not discuss here as these results were produced by coauthors and, while interesting, do not belong in this dissertation. We refer the reader to the long version of G. Erdélyi et al. [46] for a complete treatment of these problems including their parameterized complexity and approximability results. All the results related to the PM evaluation method are unique to this dissertation and have not been previously published.

Table 3.1 summarizes the classical complexity results for the X-Y PROBABILISTIC LOBBYING PROBLEM, where $\mathrm{X} \in \{\mathrm{MB}, \mathrm{IB}, \mathrm{VB}\}$ and $\mathrm{Y} \in \{\mathrm{SM}, \mathrm{AM}, \mathrm{PM}\}$. Most of the results show containment and hardness for certain complexity classes. However, in some cases we have not been able to show hardness for certain problems. In these cases we note the containment ($\in$) of the problem.

Complexity of Evaluation Methods

To begin our results we need to understand the complexity of evaluating a vote under our defined evaluation criteria. If it is computationally hard to evaluate the outcome of a vote, then it will necessarily be computationally hard to find sufficient bribery schemes for that voting system. Two of our evaluation criteria are easily computable from observation. SM requires only the investigation of each entry in the P matrix while AM requires computing a simple average. However, the evaluation procedure for PM is not so straightforward. Theorem 3.1.7 provides a polynomial time algorithm for computing the result of an election evaluated with the PM procedure.

Theorem 3.1.7 *Evaluating a winning scenario probability for* PM *is in P.*

Proof. We define a dynamic programming algorithm[4] which runs in time $\mathcal{O}(m^2)$ for a single issue, where m is the number of voters. We compute the probability $E_{i,x}$ that exactly i of the first x voters have voted "yes" on an issue. We build a table consisting of i rows and x columns called E and start indexing at $E_{1,1}$. We will create one table for each issue, r_j. We assume that P_i is the probability that voter v_i will vote "yes."

```
$E_{1,1} := P_1$
**for** $x := 1$ to $m-1$ **do**
    $E_{1,x+1} = E_{1,x} \cdot (1 - P_{x+1}) + P_{x+1} \cdot \Pi_{j=1}^{x}(1 - P_j)$
**end for**
```

[4] We note that a similar proof appears in [71].

```
for $i := 2$ to $m-1$ do
  for $k := 1$ to $m-1$ do
    $E_{i,x+1} = E_{i-1,x} \cdot P_{x+1} + E_{i,x} \cdot (1 - P_{x+1})$
  end for
end for
```

To determine whether the final probability that r_j passes exceeds t, we need to sum all elements of E such that $i > m/2$ and $x = m$. We compute the winning probability for each issue r_j with $1 \leq j \leq n$; the probability that the entire agenda passes is the product of the probabilities that each issue passes. The running time of the algorithm is $\mathcal{O}(m^2 \cdot n)$. ❑

Basic Probabilistic Lobbying Problem

We proceed by investigating each bribery method in turn. In some cases, multiple bribery methods can leverage the same proof of complexity. The first case we investigate is the MB method.

Theorem 3.1.8 MB-SM-PLP *is in P.*

Proof. The aim is to win all referenda. For each voter v_i and referendum r_j, $1 \leq i \leq m$ and $1 \leq j \leq n$, we can compute in polynomial time the amount $b(v_i, r_j)$ The Lobby has to spend to turn the favor of v_i in the direction of The Lobby (beyond the given threshold t). In particular, set $b(v_i, r_j) = 0$ if voter v_i would already vote according to the agenda of The Lobby. For each issue r_j, sort $\{b(v_i, r_j) \mid 1 \leq i \leq m\}$ non-decreasingly, yielding a sequence $b_1(r_j), \ldots, b_m(r_j)$ such that $b_k(r_j) \leq b_\ell(r_j)$ for $k < \ell$. To win referendum r_j, The Lobby must spend at least $B(r_j) = \sum_{i=1}^{\lceil (m+1)/2 \rceil} b_i(r_j)$ dollars. Hence, all referenda can be won if and only if $\sum_{j=1}^{n} B(r_j)$ is $\leq B$, the given bribery budget. ❑

Note that the time needed to execute the algorithm given in the previous proof can be bounded by a polynomial of low order. More precisely, if the input consists of m voters,

n referenda, and discretization level k, then $\mathcal{O}(n \cdot m \cdot k)$ time is sufficient to compute each $b(v_i, r_j)$. Having these values, $\mathcal{O}(n \cdot m \cdot \log(m))$ time is sufficient for the sorting phase. The sums can be computed in time $\mathcal{O}(n \cdot m)$. (Note that the time analysis can still be improved; however, a rough estimate of the computation time needed is enough to establish Theorem 3.1.8.)

Theorem 3.1.9 MB-AM-PLP *is in P.*

Proof. Let (P, C_P, B, t) be a given MB-AM-PLP instance, where $P \in \mathbb{Q}^{m \times n}_{[0,1]}$, C_P is a cost matrix, B is The Lobby's budget, and t is a given threshold. Let k be the discretization level of P, i.e., the interval is divided into $k+1$ steps of size $1/(k+1)$ each. For $j \in \{1, 2, \dots, n\}$, let d_j be the minimum cost for The Lobby to bring referendum r_j into line with the jth entry of its target vector 1^n. If $\sum_{j=1}^{n} d_j \leq B$, then The Lobby can achieve its goal that the votes on all issues pass.

We show that, for a fixed j, we can, in polynomial time, compute d_j. Therefore, the decision problem of whether The Lobby can afford to bring all referenda into line with its target vector is also in P. We compute d_j by dynamic programming. The aim is to have $\sum_{i=1}^{m} p_{i,j}/m \geq t$, i.e., $\sum_{i=1}^{m} p_{i,j} \geq mt$. Recall that $p_{i,j} \cdot (k+1)$ always gives an integer. Define $c_j = (k+1)(mt - \sum_{i=1}^{m} p_{i,j})$. This is the overall number of confidence steps The Lobby has to buy to win referendum r_j. Note that c_j is polynomial in the size of the input, as is mtk. We define $T[l, s]$ to be the minimum cost of raising $(k+1)(\sum_{i=1}^{m} p_{i,j})$ by value $s \leq c_j$ using only microbribes to the first l voters. Notice, $T[l, 0] = 0$. Let $q_{i,j}$ be the integer with $c_{i,j}[q_{i,j}] = 0$. Hence, for κ with $0 \leq \kappa < q_{i,j}$, we find $c_{i,j}[\kappa] = --$, and for κ with $q_{i,j} < \kappa \leq k+1$, we have $c_{i,j}[\kappa] > 0$. This means that $q_{i,j}/(k+1) = p_{i,j}$. As the maximum number of confidence steps we can gain by bribing voter i is $k+1-q_{i,j}$, we obtain

$$T[1, s] = \begin{cases} \infty & \text{if } s > k+1-q_{1,j}, \\ c_{1,j}[q_{1,j}+s] & \text{otherwise.} \end{cases}$$

Based on this initialization, we can compute:

$$T[l,s] = \min\left\{T[l-1,s-q] + c_{l,j}[q_{l,j}+q] \mid 0 \leq q \leq \min\{s, k+1-q_{l,j}\}\right\}.$$

In particular, $q = 0$ covers the case when no money is spent on voter l, as $c_{l,j}[q_{l,j}] = 0$. Thus, each of the polynomially many entries, $T[l,s]$, can be computed in polynomial time. In particular, $T[m,c_j] = d_j$ can be computed in polynomial time. ❑

We have not been able to prove exact lower bounds for some bribery methods in conjunction with the PM method. However, we can establish upper bounds on MB and IB under the PM criteria.

Theorem 3.1.10 MB-PM-PLP *and* IB-PM-PLP *are in NP.*

Proof. Using a standard guess and check algorithm we can show that MB-PM-PLP and IB-PM-PLP are in NP. Given a set of bribery actions, we can verify if the resulting P matrix is sufficient to pass The Lobby's agenda using the algorithm shown in Theorem 3.1.7. ❑

Though we do not know the exact lower bound, we can show an algorithm that is pseudo-polynomial with respect to the size of the budget.

Theorem 3.1.11 *There is a bribery algorithm for* MB-PM-PLP *that runs in polynomial time with respect to B, n, and m.*

Proof. Since the evaluation method PM evaluates each referendum separately we will show a dynamic programming algorithm for a single issue. We can then apply this algorithm for all m issues.

Our goal is to raise above t the probability that a given referendum r will pass as cheaply as possible when only a subset of the voters is voting. Given bribes to the subset we can define a dynamic programming algorithm that incorporates each voter $x \in \{v_1, \ldots, v_n\}$ in turn until we find the maximum probability of r passing using the least amount of our budget.

Consider an instance of MB-PM-PLP. We compute $F_x = \Pi_{j=1}^{x} p_j$, the probability that the single referendum will pass assuming that the voters $v_{x+1}, \ldots, v_n$ deterministically vote for the referendum. Without loss of generality, we assume that B is bounded by the sum of all possible bribes.

Our goal is to find the minimum cost to make $F_n > t$. We do this by finding the maximum value we can raise each F_x to, given each value $b \leq B$. This search is achieved using dynamic programming to build a $(n+1) \times B$ table D. Filling in each entry in the table takes polynomial time, so the problem is in P if the budget, B, is polynomial in the size of the input (e.g., B is input in unary or bribes are paid in dollar bills). Otherwise, the algorithm is pseudo-polynomial.

The table entry $D[n,b]$ is the maximum we can make F_n via bribes to voters $v_1, \ldots, v_n$, within budget b. Initialize $D[0,b] = 1$ for all b.

Intuitively, we bring each voter $x \in \{v_1, \ldots, v_n\}$ online one at a time. When we incorporate a new voter v_{x+1} we may or may not require a bribe to this new voter. The maximum probability (of r passing) obtainable either involves a bribe to v_{x+1}, and whatever bribes are needed with the remaining money, or does not require a bribe to v_{x+1}. This gives us the dynamic programming update.

A bribe to voter v_{x+1} (to vote for r) changes p_{x+1} to some new value p'_{x+1}. We can compute F_{x+1} for the given bribe by $F_{x+1} = F_x \cdot p'_{x+1}$.

Let $h = \max\{D[x, b-d] \cdot p'_{x+1}\}$ where d is the cost of increasing p_{x+1} to p'_{x+1}. We set $D[x+1,b] = \max\{D[x,b] \cdot p_{x+1}, h\}$. Since there are at most $k+1$ bribes to consider for each v_x, computing $\{D[x+1,b] : 0 \leq b \leq B\}$ takes $\mathcal{O}(k \cdot n \cdot B)$ operations. This is polynomial in k, n, and B. The algorithm runs in polynomial time with respect to $|B|$ and n.

We can apply this algorithm independently to all m issues. We can do this because we must pass all issues and this dynamic programming algorithm finds the minimum cost for each referendum. Since spending any less than the minimum on a per issue basis will not achieve The Lobby's agenda we can consider each issue independently. Therefore, for any number of issues, the algorithm runs in $\mathcal{O}(k \cdot n \cdot B \cdot m)$ operations. ❑

If we define PM slightly differently, so that a win happens if the probability of the scenario in which **all** referenda pass is $> t$ (instead of currently defined as each individual referenda is $> t$) we can still use the algorithm in Theorem 3.1.11. We create an equivalent instance with only one issue and add $n \times m$ unique voters. We label the voters as $v_i r_j$ with $1 \leq i \leq n$ and $1 \leq j \leq m$ and combine them all into one overarching issue. The Lobby must pass this overarching issue in order for a win.

Theorem 3.1.12 IB-SM-PLP *and* IB-AM-PLP *are in P.*

Proof. We prove that IB-SM-PLP is in P; the proof for IB-AM-PLP is analogous.

In IB-SM-PLP we are required to influence issues, not individual voters. For each issue r_j, we determine if it will pass by counting the number of voters whose probability is above the threshold t. If this number of voters is $> \lfloor n/2 \rfloor$ then this issue will pass and we do not need to determine any bribery actions.

Otherwise, we must determine the minimum cost to bring r_j to passing (bribing enough voters). For issue r_j that is not currently passing, we count the minimum number s of voters that need to be bribed. We split the voters into two groups: Y are the voters whose probability of voting "yes" is $> t$ and X is the set of voters whose probability of voting "yes" is $\leq t$. We then number the set X from cheapest to most expensive according to how much it would cost to bring the probability that x_i votes "yes" above t.

We then select voter x_s and investigate their bribery price to elevate our referendum to a "yes." Since the bribe will be evenly split across all voters we need to spend n times the cost of bribery for voter x_s in order to have a majority on the issue. We repeat this process for every issue.

After we have computed this value for all issues we compute the total amount needed, and compare to B. If the amount to spend is $\leq B$ then we accept, otherwise we reject. ❑

In order to determine the complexity of variants of the VB method, we need to formally introduce the Optimal Lobbying Problem (OL) from Christian et al. [24]. We state this problem in the standard format for parameterized complexity:

Name: OPTIMAL LOBBYING (OL)

Given: An $m \times n$ matrix E and a $0/1$ vector $\vec{Z}$ of length n. Where each row of E represents a voter, each column represents an issue, and $\vec{Z}$ represents The Lobby's target outcome.

Parameter: A positive integer b (representing the number of voters to be influenced).

Question: Is there a choice of b rows of the matrix (i.e., of b voters) that can be changed such that in each column of the resulting matrix (i.e., for each issue) a majority vote yields the outcome targeted by The Lobby?

Christian et al. [24] proved that this problem is W[2]-complete by a reduction from k-DOMINATING SET to OL (showing the lower bound) and from OL to INDEPENDENT-k-DOMINATING SET (showing the upper bound). In particular, this implies NP-hardness of OL. The following result focuses on the classical complexity of VB-SM-PLP and VB-AM-PLP.

To employ Christian et al.'s W[2]-hardness result [24], we show that OL is a special case of VB-SM-PLP and thus (parameterized) polynomial-time reduces to VB-SM-PLP. This reduction is parameter preserving for k and shows that VB-SM-PLP is W[2]-hard. Since the original reduction for OL was from INDEPENDENT-k-DOMINATING SET to show an upper bound, OL is also NP-hard. It is this NP-hardness that we make use of in these proofs. Analogous arguments apply to VB-AM-PLP.

Theorem 3.1.13 VB-SM-PLP *and* VB-AM-PLP *are NP-complete.*

Proof. Membership in NP is easy to see for both VB-SM-PLP and VB-AM-PLP.

We prove that VB-SM-PLP is NP-hard by reducing OL to VB-SM-PLP. We are given an instance $(E,\vec{Z},b)$ of OL, where E is a $m \times n$ $0/1$ matrix, b is the number of votes to be edited, and $\vec{Z}$ is the agenda for The Lobby. Without loss of generality, we may assume that $\vec{Z} = 1^n$ (see Section 3.1.1).

We construct an instance of VB-SM-PLP consisting of the given matrix $P = E$ (a "degenerate" probability matrix with only the probabilities 0 and 1), a corresponding cost matrix C_P, a target vector $\vec{Z} = 1^n$, and a budget B. C_P has two columns (i.e., we have $k = 0$, since the problem instance is deterministic, see Section 3.1.1), one column for probability 0 and one for probability 1. All entries of C_P are set to unit cost.

The cost of increasing any value in P is n, since bribes are distributed evenly across issues for a given voter. We want to know whether there is a set of bribes of cost at most $b \cdot n = B$ such that The Lobby's agenda passes. This holds if and only if there are b voters that can be bribed so that they vote uniformly according to The Lobby's agenda and that is sufficient to pass all the issues. Thus, the given instance $(E,\vec{Z},b)$ is in OL if and only if the constructed instance $(P,C_P,\vec{Z},B)$ is in VB-SM-PLP, which shows that OL is a polynomial-time recognizable special case of VB-SM-PLP, and thus VB-SM-PLP is NP-hard.

Note that for the construction above it does not matter whether we use the strict-majority criterion (SM) or the average-majority criterion (AM). Since the entries of P are 0 or 1, we have $\overline{p_j} > 0.5$ if and only if we have a strict majority of ones in the jth column. Thus, VB-AM-PLP is NP-hard too. ❑

We can also extend this proof for VB with the PM evaluation criteria.

Corollary 3.1.14 VB-PM-PLP *is NP-complete.*

Proof. The proof of Theorem 3.1.13 shows a reduction of OL to VB-AM-PLP-WIW and VB-SM-PLP-WIW. This reduction is independent of the evaluation criterion since

Table 3.2: Complexity results for X-Y PROBABILISTIC LOBBYING PROBLEM WITH ISSUE WEIGHTING, where X $\in \{\text{MB}, \text{IB}, \text{VB}\}$ and Y $\in \{\text{SM}, \text{AM}, \text{PM}\}$

Problem	Classical Complexity	Theorem or Corollary
MB-SM-PLP-WIW	NP-complete	Thm. 3.1.15
MB-AM-PLP-WIW	NP-complete	Thm. 3.1.15
MB-PM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-SM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-AM-PLP-WIW	NP-complete	Thm. 3.1.15
IB-PM-PLP-WIW	NP-complete	Thm. 3.1.15
VB-SM-PLP-WIW	NP-complete	Thm. 3.1.16
VB-AM-PLP-WIW	NP-complete	Thm. 3.1.16
VB-PM-PLP-WIW	NP-complete	Thm. 3.1.16

we create deterministic instances of OL and therefore we can extend it to an instance of VB-PM-PLP.

This shows that VB-PM-PLP is NP-complete. ❑

Probabilistic Lobbying with Issue Weighting

Table 3.2 summarizes our results for X-Y-PLP-WIW, where X $\in \{\text{MB}, \text{IB}, \text{VB}\}$ and Y $\in \{\text{SM}, \text{AM}, \text{PM}\}$. The most interesting observation is that introducing issue weights raises the complexity from P to NP-completeness for all cases of microbribery and issue bribery (though it remains the same for voter bribery). Additional results shown by G. Erdélyi [46] indicate that these NP-complete problems are fixed-parameter tractable and, in some cases, admit a FPTAS.

To begin we first need to introduce the well known NP-complete problem KNAPSACK [65].

Name: KNAPSACK

Given: given a set of objects $U = \{o_1, \ldots, o_n\}$ with weights $w : U \to \mathbb{N}$ and profits $p : U \to \mathbb{N}$, and $W, P \in \mathbb{N}$.

Question: Is there a subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in I} w(o_i) \leq W$ and $\sum_{i \in I} p(o_i) \geq P$.

Theorem 3.1.15 *The problems* MB-SM-PLP-WIW*,* MB-AM-PLP-WIW*,* MB-PM-PLP-WIW*,* IB-SM-PLP-WIW*,* IB-AM-PLP-WIW*, and* IB-PM-PLP-WIW *are NP-complete.*

Proof. Membership in NP can be seen for each problem X-Y-PLP-WIW, $X \in \{MB, IB\}$ and $Y \in \{SM, AM, PM\}$ through a guess and check algorithm.

To prove that MB-SM-PLP-WIW is NP-hard, we give a reduction from KNAPSACK. Given a KNAPSACK instance (U, w, p, W, P), create a MB-SM-PLP-WIW instance with $k = 0$ and only one voter, v_1, where for each issue, v_1's acceptance probability is either zero or one. For each object $o_j \in U$, create an issue r_j such that the acceptance probability of v_1 is zero. Let the cost of raising this probability on r_j be $c_{1,j}(1) = w(o_j)$ and let the weight of issue r_j be $w_j = p(o_j)$. Let The Lobby's budget be W and its objective value be $V = P$. By construction, there is a subset $I \subseteq \{1, \dots, n\}$ with $\sum_{i \in I} w(o_i) \leq W$ and $\sum_{i \in I} p(o_i) \geq P$ if and only if there is a subset $I \subseteq \{1, \dots, n\}$ with $\sum_{i \in I} c_{1,i} \leq W$ and $\sum_{i \in I} w_i \geq V$.

As the reduction introduces only one voter, there is no difference between the bribery methods MB and IB, and no difference either between the evaluation criteria SM, AM, and PM. Hence, the above reduction works for all six problems. ❑

Turning to voter bribery with issue weighting an immediate consequence of Theorem 3.1.13 is that VB-SM-PLP-WIW, VB-AM-PLP-WIW, and VB-PM-PLP-WIW are NP-hard, since they are generalizations of VB-SM-PLP, VB-AM-PLP, and VB-PM-PLP-WIW. Again, membership in NP can also be seen for the issue weighted problems

Corollary 3.1.16 VB-SM-PLP-WIW*,* VB-AM-PLP-WIW*, and* VB-PM-PLP-WIW *are NP-complete.*

3.1.7 Observations

We have studied four lobbying scenarios in a probabilistic setting, both with and without issue weights. Among these, we identified problems that can be solved in polynomial time and problems that are NP-complete. For the case of weighted issues we find that all problem variants are, in fact, strongly NP-hard; their hardness does not depend on whether the numbers in their inputs are encoded in unary or in binary. In some cases not discussed in this presentation of the results, we find problems that are fixed-parameter tractable problems, and problems that are hard (namely, W[2]-complete or W[2]-hard) in terms of their parameterized complexity with suitable parameters. The additional results also investigate the approximability of hard probabilistic lobbying problems (without issue weights) and obtain both approximation and inapproximability results. A complete treatment of these results can be found in the paper by G. Erdélyi et al. [46].

As a general statement, this section shows the addition of uncertainty into the reasoning process increases the computational complexity. While this is not true in all cases, a direct comparison is difficult to classify. Unweighted, deterministic bribery is computationally tractable, while almost all weighted variants are computationally hard [52]. We obtain mixed results in this section. This can be attributed, in large part, to the particular modeling choices. We have provided a mix of models in an attempt to identify where, exactly, the problem becomes hard. We continue this quest for a clear dividing line in the next section.

3.2 Sports Tournaments and Ranking Problems

Sports competitions are common forms of entertainment and recreation around the world. In most sports contests both observers and players have some notion of which competitors are favored over others. Many individuals, including some players, wager vast sums of money on the outcomes of particular games and tournaments. A quick Google search reveals dozens of players, coaches, referees, and judges convicted of manipulating the outcome of sports competitions through match fixing, point shaving, and outright cheating. Additionally many websites (such as `www.kenpom.com`) produce and publish in depth statistics for not only overall team win/loss predications, but also predictions for individual player stats on a per game basis. It is a world of probabilities and manipulation.

We use sports tournaments as a motivating example of other domains in which bribery [52] and coalitional manipulation [33] can undermine the integrity of competition. Tournaments and single winner elections, when the set of candidates and the set of voters are equivalent, are used in many domains including self-organization of ad-hoc wireless sensor networks, where leaders are elected to delegate work or act as central routing nodes [121], and multi-criteria decision making, where page rankings are sometimes determined by links from the set of pages under consideration [16]. In addition to these important applications of tournaments, there has been recent empirical research in political science and sociology revealing that, in the United States, voter preferences in political elections can be significantly affected by apparently irrelevant events, specifically sports tournaments [73].

In this section we study three different types of sports tournaments.

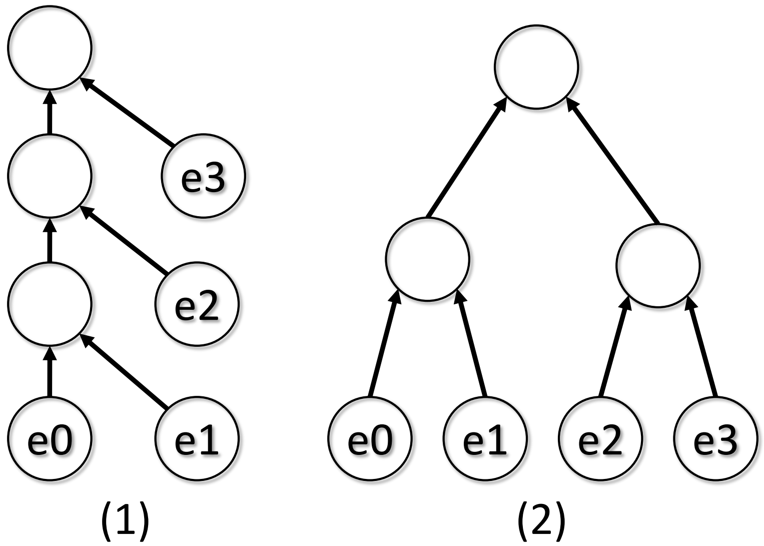
Cup Tournament: A single-elimination competition (or knockout tournament [128]) over a complete binary tree where each entrant[5] plays a sequence of matches head-to-head; the winner is the entrant that is left undefeated. The United States' men's and women's NCAA Basketball Tournaments and most tennis majors fall into this category.

Round Robin tournament: A competition where each entrant competes against every other entrant and earns a points for each victory; the winner is the entrant with the most points. The group play round of the FIFA World Cup falls into this category.

Challenge or Caterpillar Tournament: A series of matches where the winner of each match plays the next entrant in increasing order of rank; the winner is the entrant

[5] We use the term entrant in this section because we can imagine a tournament made up of individuals or teams.

Figure 3.1: Example of (1) A challenge tournament and (2) a cup tournament. The winner is the entrant who reaches the top node.



who wins the final match. Boxing titles and some PBA bowling competitions use this type of tournament.

Figure 3.2 illustrates the difference between cup and challenge tournaments. These types of sporting events correspond to the voting rules: cup for cup tournaments, Copeland for round robin tournaments, and linear balloting for challenge tournaments. We refer the reader to Section 2.2.1 or to Arrow et al. [3] for a more complete treatment of voting rules. In tournaments and sporting events there are several natural questions which arise, such as: "What are the odds my preferred entrant wins?" and "Does my preferred entrant have a chance of winning?"

The classical notation of manipulation, introduced by Bartholdi et al. [7], has been extensively studied in the deterministic case. Conitzer et al. [33] studied some manipulation problems in stochastic settings, however, many of their NP-completeness results break down in the setting under study here because the reductions require the ability to add voters that are not in the candidate set. There are also results relating to manipulation under deterministic information for cup [33] and Copeland [55]. Likewise, the bribery problem for elections, introduced by Faliszewski et al. [52], has been well studied for voting rules under deterministic information. Again, many of these results do not transfer; the hardness reductions require the ability to introduce an unrestricted number of non-candidate voters.

There has been significant work on the schedule control problem for cup or knockout tournaments: in the deterministic case by Lang et al. [81] (for the similar problem of sequential majority voting), Vu et al. [128], and Williams [132]; in the stochastic case by Hazon et al. [72]. In addition to the work on the control problem, there is work from Russell and Walsh about the complexity of coalitional manipulation in sports tournaments [113]. In Russell and Walsh's model, entrants may be a part of a coalition of manipulators that can choose to lose their matches. There is also an extensive body of work on the elimination problem for sports tournaments. In the elimination problem, the probability that team i beats team j is known, and the problem is to find the probability that a team is eliminated given a sports season or playoff schedule [68, 76].

There has been surprisingly little work on stochastic models of tournaments; perhaps the closest notion is that of a *possible winner* — a notion intrinsic to reasoning under uncertainty introduced by Konczak and Lang [78], with further study by Lang et al. [81], and others. Recent studies of the complexity of computing possible winners include those by Lang et al. [81], Faliszewski et al. [4], and Xia et al. [139]. These papers address complexity questions when voters are defined by their (possibly incomplete) preference profiles over a set of outcomes, but not probabilities of winning.

In Section 3.2.1 we formally describe our model of reasoning about bribery in tournaments where entrants have probabilities of winning and losing. Section 3.2.2 formally states the six decision problems we study in this domain for all three tournament types. Section 3.2.3 details our complexity results and ends with a few observations about the change in reasoning complexity when models move from deterministic tournaments to models of uncertain information.

3.2.1 Model Definition

We begin by defining a model with which to reason about sports tournaments and other head-to-head competitions. The details of the model closely follow the model proposed in Section 3.1.1 for plurality elections [45].

Consider a tournament with n entrants $\{e_1, \dots, e_n\}$. Note that what distinguishes a tournament from a voting rule here is that, for a given election with separate sets of candidates and voters, we require the set of candidates to be exactly the set of voters. Let $\mathbb{Q}^{n\times n}_{[0,1]}$ be the set of $n \times n$ matrices over $[0,1] \cap \mathbb{Q}$. Let $p_{i,j}$ denote the probability that entrant i will defeat entrant j. Let $P = [p_{i,j}] \in \mathbb{Q}^{n\times n}_{[0,1]}$. We require that $p_{i,j} + p_{j,i} = 1$. We choose $k \in \mathbb{N}_+$ to discretize the interval [0,1] (we call this the *discretization level* of the problem). That is, we let each $p_{i,j}$ be in $\{0, 1/(k+1), 2/(k+1), \dots, k/(k+1), 1\}; i, j \in \{1, \dots, n\}$. Taking $k = 0$ gives us the deterministic case where each game is either won or lost with certainty.

We define C_P as the **discrete price table**. C_P has n^2 rows (indexed by pairs i, j) and $k+2$ columns. For each entry in the table we have a positive integer ($\mathbb{Z}^+$) value representing the cost to lower $p_{i,j}$ to a given probability. We assume that all entrants will compete to the best of their abilities and we cannot increase an entrant's probability of winning a particular match. Therefore we designate these entries as $--$. We also require that $c_{i,j}(p_{i,j}) = 0$. That is, it does not cost anything to have an entrant compete at its highest level. The entries for $c_{i,j}$ must also be monotone decreasing; payment of $c_{i,j}(l)$ changes the probability that i beats j to $l/(k+1)$. Bribery must be performed in one fell swoop, before the first game is played. Outside actors may not have access to the entrants after the start of the competition (i.e., the tournament is taking place in a remote or tightly controlled location) so all bribery must be done beforehand.

We are also given a threshold t and an integer-valued budget B. For cup and challenge tournaments we are also given a tree T with entrants labeled on the leaves to represent the order of the matches. Let $Pr[e_i|P,T]$ denote the probability that e_i wins the tournament defined by schedule T and probability matrix P.

Example 3.2.1 *Consider the following example with $k = 4$ (so there are $k+1$ = 5 possible probability configurations) and $n = 4$ entrants. We use this example setup as a running example for the rest of the chapter. These parameters for k and n are given implicitly by the probability matrix P and cost matrix C_P.*

$P =$

	e_1	e_2	e_3	e_4
e_1	$--$	0.75	0.50	0.25
e_2	0.25	$--$	0.25	0.25
e_3	0.50	0.75	$--$	0.25
e_4	0.75	0.75	0.75	$--$

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{1,2}$	100	40	20	0	$--$
$c_{1,3}$	30	25	0	$--$	$--$
$c_{1,4}$	200	0	$--$	$--$	$--$
$c_{2,1}$	10	0	$--$	$--$	$--$
$c_{2,3}$	10	0	$--$	$--$	$--$
$c_{2,4}$	10	0	$--$	$--$	$--$
$c_{3,1}$	120	75	0	$--$	$--$
$c_{3,2}$	200	100	50	0	$--$
$c_{3,4}$	300	0	$--$	$--$	$--$
$c_{4,1}$	300	200	100	0	$--$
$c_{4,2}$	300	200	100	0	$--$
$c_{4,3}$	400	300	200	0	$--$

In this section we focus on the following three types of tournaments. In each case we assume that no game ends in a tie and we have a unique winner for every tournament. In the deterministic case a scoring model which includes ties or does not normalize to certain forms for round robin tournaments and Copeland elections can have significant effects on the complexity of evaluation, manipulation, and bribery [55, 76].

Challenge Tournament: In a challenge tournament the entrants are ordered $e_1, e_2, \ldots, e_n$. In the first match, e_1 plays e_2. In the second match, the winner plays e_3, and so on. The winner of the last match wins the tournament.

Cup: In a cup tournament we are given a complete binary tree T with entrants labeled on the leaves. Each internal node is decided by the competition between the two entrants on the level below. The winner is the entrant who reaches the top node of the tree.

Round Robin: In a round robin tournament (Copeland Scoring) each entrant plays every other entrant exactly once. In each match each entrant receives 1 point for a win and 0 points for a loss. The winner of the tournament is the entrant with the maximum number of points. If more than one entrant has the same maximum score, we employ a lexicographic tie-breaking scheme where e^* always comes first.

3.2.2 The Probabilistic Tournament Bribery Problem

With this model we can now define our problem and a set of related decision problems for $Y \in \{$Challenge Tournament (CT), Cup, Round-Robin (RR)$\}$.

Name: Y - PROBABILISTIC TOURNAMENT BRIBERY PROBLEM (Y-PTBP)

Given: A probability matrix $P \in \mathbb{Q}^{n\times n}_{[0,1]}$ describing the entrants e_i in the tournament with some preferred entrant e^* and (where necessary) cost matrix C_P, threshold t, budget B, set of manipulators $M \subseteq E$, and ordering T.

Questions: We define the following 6 decision problems on the above information:

Evaluation: Is the probability that e^* wins Y (the sum of the probabilities of the futures with e^* a winner) above t ?

Pos-Win: Is e^*'s probability of winning the tournament above 0?

Pos-Win-\$: Can we raise e^*'s probability of winning Y above 0 by bribing specific entrants according to C_P and not exceeding the budget B?

Constructive Coalitional Manipulation (CCM): Can we raise e^*'s probability of winning Y above t by strategically setting all the probabilities associated with a subset M (a *coalition*) of the entrants.

Constructive Bribery: Can we raise e^*'s probability of winning Y above t by bribing specific entrants according to C_P and not exceeding the budget B?

Exact: Can we raise e^*'s probability of winning the tournament above t by bribing specific entrants according to C_P and spending exactly B?

Example 3.2.2 *Recall our running example matrices, shown below.*

$P =$

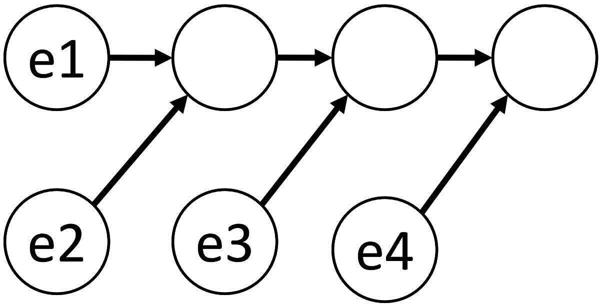
	e_1	e_2	e_3	e_4
e_1	−−	0.75	0.50	0.25
e_2	0.25	−−	0.25	0.25
e_3	0.50	0.75	−−	0.25
e_4	0.75	0.75	0.75	−−

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{1,2}$	100	40	20	0	−−
$c_{1,3}$	30	25	0	−−	−−
$c_{1,4}$	200	0	−−	−−	−−
$c_{2,1}$	10	0	−−	−−	−−
$c_{2,3}$	10	0	−−	−−	−−
$c_{2,4}$	10	0	−−	−−	−−
$c_{3,1}$	120	75	0	−−	−−
$c_{3,2}$	200	100	50	0	−−
$c_{3,4}$	300	0	−−	−−	−−
$c_{4,1}$	300	200	100	0	−−
$c_{4,2}$	300	200	100	0	−−
$c_{4,3}$	400	300	200	0	−−

Consider a challenge tournament with the entrants ordered on T as illustrated in Figure 3.2.2

Figure 3.2: Tournament graph (T) for Example 3.2.2.



For this graph we can evaluate the six questions presented in above. We assume that $e^* = e^1$ *and we enumerate the other necessary parameters for the questions that require them.*

Evaluation: *For the decision problem we require a threshold, here we take* $t = 0.08$. *For a caterpillar tournament with* e^* *in the first position, computing the exact probability of winning is* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.25 = 0.09375$. *Since this is* $> t$ *this example is a "yes" instance.*

Pos-Win: *For the decision problem for pos-win we only need to examine the result of the evaluation procedure. Since* $Pr[e_1|P,T] > 0$ *then this example is a "yes" instance.*

Pos-Win-\$: *For our running example* $Pr[e_1|P,T] > 0$ *without requiring any bribes and therefore is also a "yes" instance for pos-win-\$.*

Constructive Coalitional Manipulation (CCM): *For this question let* $M = \{e_4\}$ *and* $t = 0.15$. *Since we are trying to raise* $Pr[e_1|P,T] > 0.15$ *the best thing to do is to set* $p_{1,4} = 0$. *This way the manipulator,* e_4, *loses to the preferred entrant with certainty. We can then evaluate* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 1.0 = 0.375$. *Since* $Pr[e_1|P,T] > 0.15$ *this is a "yes" instance.*

Table 3.3: Complexity results for the PROBABILISTIC TOURNAMENT BRIBERY PROBLEM. In some cases we have been unable to provide lower bounds, in these cases we note our upper bound results ($\in$).

	Challenge	Cup	Round Robin
Evaluation	P (Thm. 3.2.3)	P (Thm. 3.2.10)	$\in$ #P (Thm. 3.2.15)
Pos-Win	P (Thm. 3.2.4)	P (Thm. 3.2.11)	P (Cor. 3.2.18)
Pos-Win-$	P (Thm. 3.2.5)	P (Thm. 3.2.12)	P (Thm. 3.2.17)
CCM	P (Thm. 3.2.6)	$\in$ NP (Thm. 3.2.13)	$\in \mathrm{NP}^{\mathrm{PP}}$ (Thm. 3.2.16)
Cons. Bribery	$\in$ NP (Thm. 3.2.8)	$\in$ NP (Thm. 3.2.13)	$\in \mathrm{NP}^{\mathrm{PP}}$ (Thm. 3.2.16)
Exact	NP-C (Thm. 3.2.9)	NP-C (Thm. 3.2.14)	$\in \mathrm{NP}^{\mathrm{PP}}$ (Thm. 3.2.16)

Constructive Bribery: *For this question let* $t = 0.15$ *and* $B = 100$*. By investigation we see there are only two meaningful options for bribes. We can bribe* e_4 *so that* $p_{1,4} = 0.5$ *or we can bribe* e_2 *and* e_3 *so that* $p_{1,2} = 1.0$ *and* $p_{1,3} = 0.75$*. Using the evaluation formula again we see that with the first option of bribes we have* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.50 = 0.1875$ *with the second option of bribes we have* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 1.0 \times 0.75 \times 0.25 = 0.1875$*. Both of these options give us the same value for* $Pr[e_1|P,T]$ *and, since both options are* > 0.15*, this is a "yes" instance.*

Exact: *For this question let* $t = 0.15$ *and* $B = 100$*. By investigation we see there are only two meaningful options for bribes and only one option that exactly spends our budget. Therefore, we bribe* e_4 *so that* $p_{1,4} = 0.5$*. Using the evaluation formula again we see that* $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.75 \times 0.5 \times 0.50 = 0.1875$*. Since* $Pr[e_1|P,T] > 0.15$ *and we have spent exactly our budget this is a "yes" instance.*

3.2.3 Results

We proceed through the results in order of tournament type. This allows us to present the results in a coherent fashion as, in many cases, the proofs and algorithms build on one another.

Challenge Tournament

We begin by investigating the evaluation problem for challenge tournaments. It is important to establish that we have a polynomial time winner determination scheme, otherwise, the bribery results would, by definition, be computationally hard.

Theorem 3.2.3 *Evaluation for* CT-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the winning probability of any entrant in an instance of CT-PTBP. Let the entrants be $\{e_1, \ldots, e_n\}$. Suppose $e^* = e_k$ and $1 \leq k \leq n$. Let $z(i, j)$ denote the probability that e_i wins match j.

```
Let $z(0,0) = 1$.
**for** $i = 1$ to $k$ **do**
    $u = max(1, i-1)$ {The first match $e_i$ plays.}
    $z(i,u) = \sum_{t=1}^{i-1} z(t, u-1) p_{i,t}$
    **for** $j = max(1, i-1)$ to $n-1$ **do**
        $z(i, j) = z(i, j-1) p_{i,j}$
    **end for**
**end for**
```

At the termination of the algorithm, the probability that e_i wins is equal to $z(i, n-1)$.
❑

Now that we have determined that it is computationally easy to evaluate the result of a given instance of CT-PTBP, we move our attention to the possible winner problem.

Theorem 3.2.4 *Pos-Win for* CT-PTBP *is in P.*

Proof. Run the evaluation algorithm given in Theorem 3.2.3 with $t = 0$. Note that e^* is a possible winner if and only if $Pr[e^*|P, T] > 0$. ❑

Theorem 3.2.5 *Pos-Win-\$ for* CT-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the minimum cost such that e^* has a non-zero probability of winning the tournament.

In order for entrant e^* to have a non-zero probability of winning it must be able to beat at least one possible entrant in each of e^*'s matches in the tournament graph T. We compute the minimum cost to ensure that each entrant has a non-zero probability of reaching level L in the graph.

Given P and C_P we can compute the **minimum cost matrix**, *min*\$, an integer matrix of size n^2, as follows: $min\$_{ij} = 0$ if i starts with a non-zero probability to beat j and the minimum bribe cost $min\$_{ij} = c_{ij}(1/(k+1))$ otherwise. Let L be a level in the tournament graph T starting with 0 at the bottom level. We then create a vector for each entrant, V_{e_i}, of size n with all entries initialized to 0 (the cost to get to the 0th level). Let $V_{e_i}(L)$ be the minimum cost for e_i to be a winner at level L. Without loss of generality we assume that the entrants are numbered such that e_1 and e_2 compete in the first match with the winner competing against e_3 in the second match.

```
for $L = 1$ to $n-1$ do
    Let $G = e_1, ..., e_{L+1}$.
    for $e \in G$ do
        if $e = e_{L+1}$ then
            $V_e(L) = \min_{x \in G, x \neq e}(min\$_{j,x} + V_x(L-1))$
        end if
        if $e = e_j, j \leq L$ then
            $V_e(L) = min\$_{j,L+1} + V_e(L-1)$
        end if
    end for
end for
```

At the end of execution V_{e_i} will contain the minimum cost to promote entrant i to the top level of T with a non-zero winning probability. We can compare this cost with B and either accept if $V_{e^*}(n-1) \leq B$ else reject. ❑

Recall that in the coalitional manipulation problem we are given a set $M \subseteq E$ of members of the manipulating coalition. Observe that the coalitional manipulation problem is a special case of the bribery problem in our model. Specifically, in a coalitional manipulation instance, $B = |M|$ and the cost of bribing members of M is also 1 (unit priced). The entrants $E \setminus M$ have bribery costs equal to ∞ (so they cannot be changed in this scenario).

Theorem 3.2.6 *Constructive Coalitional Manipulation for* CT-PTBP *is in P.*

Proof. Since there are no budget-related resource bounds in this problem we can assume our problem is to maximize e^*'s probability of winning by setting the entries for each $m \in M$ in the probability matrix P. We refer to the setting of M as a strategy. In this instance, C_P gives the available options of probability values for each $m \in M$ even though there are only unit prices.

There are $|M| \cdot (n - |M|)$ potential contests between coalition and non-coalition members with 2 possible strategies for each, and there are $|M| \cdot \frac{|M|-1}{2}$ contests between coalition members, with 3 possible strategies for each (a loses, b loses, both try to win), for a total of $2^{|M| \cdot (n-|M|)} 3^{|M| \cdot \frac{|M|-1}{2}}$ strategies.

Any manipulators who enter the tournament after e^* need to deterministically lose to e^* in order for e^* to win the championship. Therefore, for the rest of the proof, we focus on strategies for manipulators who enter the tournament before e^*. We construct a strategy in a step by step fashion starting with the first game that e_{n-1} participates in. We proceed in reverse match order, one game at a time, setting the strategies for each $m \in M$ against each entrant as we go.

Step 0: For each $m \in M$, if m reaches e^* then m chooses to lose. This sets the strategy against e^*.

Repeat for: $j = 1$ to $n-2$

> **Step *i*:** For each $m \in M$ that enters before e_{n-j}, suppose m reaches e_{n-j}. Set m's strategy against e_{n-j} to maximize the probability that e^* wins by evaluating the 2 or 3 possibilities. This is possible because strategies for all contests above e_{n-j} have been set. Also, if $e_{n-j} \in M$, then for each $e_i \notin M$, $i < n-j$, assuming e_i reaches e_{n-j}, pick e_{n-j}'s strategy against e_i to maximize e^*'s probability of winning.

After Step $j = n-2$ all strategies have been set for all M. Whatever the strategy is at lower levels, $Pr(e^* \text{ wins })$ is maximized by maximizing

$$q_{ij} = Pr(e^* \text{ wins } | e_i \text{ reaches } e_{n-j}). \tag{3.1}$$

Since the events "e_i reaches e_{n-j}" for different i are disjoint, the q_{ij} can be maximized independently.

We do 2 or 3 evaluations for each pair of contestants. For each we require a call to the $\mathcal{O}(n^2)$ evaluation procedure described in Theorem 3.2.3. Therefore we compute an optimal manipulation strategy in $\mathcal{O}(n^4)$. ❑

Theorem 3.2.3 gives us membership in NP for the constructive bribery problem since we can verify a bribery plan in polynomial time. Theorem 3.2.6 shows that in a situation of CCM we can achieve optimal manipulations in polynomial time. However, our results for CCM do not extend in a straightforward manner to the situation of priced bribery.

The simple answer would be to apply a *bang-for-the-buck* greedy algorithm to iteratively choose the cheapest available single bribe which results in the largest change in $Pr[e^*|P,T]/\text{COST OF BRIBE}$. However, this will not reach an optimal solution in all cases. This can be illustrated by an example of CT-PTBP. Consider the following example over four entrants.

Example 3.2.7 *Assume that $e^* = e_1$ and the tournament graph, T, is the same as shown in Figure 3.2.2 (where e_1 plays every match). Assume we have $k = 4$ and P and C_P given below with $B = 20$. For simplification, we omit the bribery cost for all matches that do not contain e^*.*

$P =$

	e_1	e_2	e_3	e_4
e_1	$--$	0.25	0.25	0.25

$C_P =$

$c_{i,j}$	0.00	0.25	0.50	0.75	1.00
$c_{2,1}$	13	13	13	0	$--$
$c_{3,1}$	25	10	10	0	$--$
$c_{4,1}$	25	10	10	0	$--$

We know that when e^* is involved in every match, the winning probability is given by: $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.25 \times 0.25 = 0.015625$. We set the cheapest bribe of entrant e_2 to change $p_{1,2} = 1.0$ with cost 15. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 1.0 \times 0.25 \times 0.25 = 0.0625$, giving a change in probability of 0.046875 for a cost of 13 or a bang-for-the-buck ratio of 0.0036058. Looking at either a bribe of e_3 or e_4 we would change $p_{1,3} = 0.75$ with a cost of 10. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.75 \times 0.25 = 0.046875$, giving a change in probability of 0.03125 for a cost of 10 or a bang-for-the-buck ratio of 0.003125. This would mean that according to the bang-for-the-buck heuristic, bribing e_2 is best single action.

However, looking at our budget we see that we can afford to bribe both $p_{1,3} = p_{1,4} = 0.75$ for a combined cost of 20. This would give $Pr[e_1|P,T] = p_{1,2} \times p_{1,3} \times p_{1,4} = 0.25 \times 0.75 \times 0.75 = 0.140625$. This is, by far, the best action and shows that a simple single action greedy algorithm would make non-optimal decisions in this case.

In order to eliminate as many complications of the tournament graph as possible, we turn our attention to the simplest case of a challenge tournament. This is a version where e^* is potentially involved in every game. In this version, CHALLENGE-THE-CHAMP-PTBP, e^* takes on all challengers and is a common sub-problem for both the general CT-PTBP and CUP-PTBP problems.

Theorem 3.2.8 *There is a Constructive Bribery algorithm for* CHALLENGE-THE-CHAMP-PTBP *that runs in polynomial time with respect to B and n.*

Proof. The Challenge-the-Champ problem assumes that $e^* = e_1$ and is in the first match. Our goal is to maximize the probability that e^* will win the tournament within budget through bribes to each of the other players in the tournament. The idea of this proof is to consider bribes on only a subset of the entrants. At each step we add an entrant $x \in \{e_2, \ldots, e_n\}$ and show a dynamic programming update that will compute the cheapest bribe for the new subset of entrants until we have considered the set of all the entrants.

In an instance of Challenge-the-Champ we compute the probability that e^* will win the first x games with $F_x = \Pi_{j=1}^{x} p_{1,j}$. Intuitively F_x tells us the probability that e^* wins the tournament considering only x of the n games. Without loss of generality, we assume that B is bounded by the sum of all possible bribes.

Our goal is to maximize F_x (within budget). We do this using dynamic programming to build a $(n+1) \times B$ table D. Filling in each entry in the table takes polynomial time, so the problem is in P if the budget, B, is polynomial in the size of the input (e.g., B is input in unary). Otherwise, the algorithm is pseudo-polynomial.

The table entry $D[n,b]$ is the maximum we can make F_n via bribes to entrants $e_1, \ldots, e_n$, within budget b. Initialize $D[0,b] = 1$ for all b.

Intuitively, we add games to the set one at time by considering each new entrant $x \in \{e_2, \ldots, x_n\}$. When we incorporate the new e_{x+1} we may or may not require a bribe to this new entrant. The maximum probability (of e^* winning) obtainable either involves a bribe to e_{x+1}, and whatever bribes are needed with the remaining money, or does not require a bribe to e_{x+1}. This gives us the dynamic programming update.

A bribe to entrant e_{x+1} (to lose to e^*) changes $p_{1,x+1}$ to some new value $p'_{1,x+1}$. We can compute F_{x+1} for the given bribe by $F_{x+1} = F_i \cdot p'_{1,x+1}$.

Let $h = \max\{D[x, b-d] \cdot p'_{1,x+1}\}$ where d is the cost of increasing $p_{1,x+1}$ to $p'_{1,x+1}$. We set $D[x+1,b] = \max\{D[i,b] \cdot p_{1,x+1}, h\}$. Since there are at most $k+1$ bribes for each e_i, computing $\{D[i+1,b] : 0 \leq b \leq B\}$ takes $\mathscr{O}(k \cdot n \cdot B)$ operations. This is polynomial in k, n, and B. The algorithm runs in polynomial time with respect to B and n. ❑

While we have been unable to show NP-hardness for any version of the constructive bribery problem, Theorem 3.2.8, with the insights listed in this section, leads to the conjecture that the problem is hard. Many well known NP-hard problem like KNAPSACK admit pseudo-polynomial time dynamic programming algorithms [65].

If we require our outside manipulator to spend exactly the allocated budget then we can show NP-completeness for challenge tournaments. Situations which require an organization to spend a budget exactly occur implicitly in many large organizations and governments. For resources other than money that could be used, such as referees in sports tournaments or canvassers in political elections, there sometimes the requirement of exact allocations.

We must introduce the SUBSET SUM PROBLEM for our next proof. The statement of the problem is from Garey and Johnson's book [65].

Name: SUBSET SUM

Given: A set $W \in \mathbb{Z}_+$, $w_1, \ldots, w_m$, and a target $S \in \mathbb{Z}_+$.

Question: Does there exist a subset $I \subseteq \{1, \ldots, m\}$ such that $\sum_{i \in I} w_i = S$?

This problem was shown to be NP-complete via a transformation from PARTITION by Karp [75]. Using this problem we can exactly classify the complexity of the exact bribery problem for CT-PTBP.

Theorem 3.2.9 *Exact Bribery for* CT-PTBP *is NP-complete.*

Proof. Membership is an immediate consequence of Theorem 3.2.3 and a guess and check algorithm.

To show NP-hardness we provide a reduction from Subset Sum: For a given Subset Sum instance we set up a challenge tournament with S players such that e^* always wins (i.e., e^* has a 100% chance of winning all games against all players) and we let $t = 0$, $B = S$, and $k = 0$. We then create the bribery cost matrix C_P with prices equal to the weights of $w_1, \dots, w_m$. In this situation bribery will have no effect and we must distribute the money exactly. We have now established a polynomial time mapping such that there will be an exact bribery if and only if there is a subset such that $\sum_{i \in I} w_i = S$. ❑

Cup Rule

We observe that many of the results for Cup-PTBP use reasoning similar to that used for CT-PTBP. We assume that each Cup-PTBP instance is a complete tree. Note that this does not limit our results; we can always pad the tree with entrants who lose to all other entrants and cannot be bribed.

Theorem 3.2.10 *Evaluation for* Cup-PTBP *is in P.*

Proof. This is a proof by construction of a polynomial time algorithm that computes $Pr[e^*|P,T]$ for all entrants. We construct a table of size $n \cdot \lceil \log_2(n) \rceil$ and use dynamic programming to compute for each $i : L_{r,i}$ where $L_{r,i}$ is the probability that entrant i advances to round r (with $r = 1$ being the first match e_i competes in). When we get to the last round, our table will contain the probability that each entrant won.

```
Let $G(r,i)$ be the entrants that $e_i$ may face in round $r$.
for $i = 1$ to $n$ do
    $L_{1,i} = 1$
end for
for $r = 2$ to $\lceil \log_2(n) \rceil$ do
    for $i = 1$ to $n$ do
        $L_{r+1,i} = L_{r,i} \cdot \sum_{x \in G(r,i)} p_{i,x} \cdot L_{r,x}$
    end for
  end for
```

At each stage, r, the numerators and denominators of the probabilities are $\mathscr{O}(k^r) \subset \mathscr{O}(k^n)$, having sizes $n\log(k)$ (where k is the discretization level). Each e_i potentially competes against each x in exactly one round, so there are exactly $n(n-1) \in \mathscr{O}(n^2)$ multiplications in total so the total time is $O(n^2(n\log(k))^2) = O(n^4\log(k)^2)$. ❑

Turning again to the possible winner problem variants we see that evaluating whether a entrant has a chance of winning is easy for cup tournaments as well.

Theorem 3.2.11 *Pos-Win for* CUP-PTBP *is in P.*

Proof. Run the evaluation algorithm given in Theorem 3.2.10 with $t = 0$. ❑

Theorem 3.2.12 *Pos-Win-\$ for* CUP-PTBP *is in P.*

Proof. We provide a polynomial time dynamic programming algorithm to compute the minimum cost such that e^* has a non-zero probability of winning the tournament.

Let the tournament graph T be a complete binary tree. Let L be the level of the binary tree starting with 0 at the bottom level. We then create a vector for each entrant, V_{e_i}, of size n with all entries initialized to 0 (the cost to get to the $0th$ level). Let $V_{e_i}(L)$ be the minimum cost for e_i to have a non-zero probability of winning at level L. We construct the **minimum cost matrix**, $min\$$, an integer matrix of size n^2, as follows: $min\$_{ij} = 0$ if i starts with a non-zero probability to beat j and the minimum bribe cost $min\$_{ij} = c_{ij}({}^{1}/_{(k+1)})$ otherwise.

```
for $L = 1$ to $\log(n)$ do
  for all $e_i \in T$ do
    Let $G(e_i, L)$ be the set of entrants in the sub-tree at level $L$ containing $e_i$.
    $V_{e_i}(L) = V_{e_i}(L-1) + \min_{x \in G(e_i,L) - G(e_i,L-1)}(min\$_{i,x} + V_x(L-1))$
  end for
```

end for

At the end of execution $V_{e_i}(\log_2(n))$ will contain the minimum cost to promote e_i to the top level of T with a non-zero winning probability. We can compare this cost with B and either accept if $V_{e^*}(\log_2 n) \leq B$ or reject. ❑

Theorem 3.2.10 immediately provides us with the following corollaries through standard guess and check algorithms.

Corollary 3.2.13 *Constructive Coalitional Manipulation and Constructive Bribery are in NP for* CUP-PTBP.

We have not been able to show lower bounds results for CCM or constructive bribery for CUP-PTBP. The result does not hold for cup tournaments because the winner within separate sub-trees can affect the winning probability of all entrants in a neighboring subtree. We can, however, straightforwardly transfer our results for the exact case.

Theorem 3.2.14 *Exact Bribery for* CUP-PTBP *is NP-complete.*

Proof. To show membership in NP we can easily check whether we have spent exactly our budget. We can nondeterministically guess a bribery scheme that utilizes the entire budget and check it using the algorithm developed in the proof of Theorem 3.2.10. Thus we have a polynomial time algorithm which requires a nondeterministic guess to come up with an exact bribery scheme.

To show NP-hardness we can use a similar reduction as the one used in Theorem 3.2.9. Recall that S is the target sum in a SUBSET SUM instance. In this construction we choose a number, j, such that $j^2 \geq |S|$. We build our cup instance with j players. We set the added players' bribery prices to be $> B$ so that these players can be safely ignored. We then use the same mapping presented in the proof of Theorem 3.2.9 to show Exact Bribery is NP-complete. ❑

Round Robin

Round Robin tournaments are of particular interest in the social choice community because of their close correspondence to the Copeland election system. Faliszewski et al. [55] provided a comprehensive study of bribery and manipulation of Copeland elections under deterministic scenarios.

In addition, many papers on sports elimination, including those by Gusfield and Martel [68] and Kerns and Paulusma [76], study sports round robin tournaments for particular competitions (FIFA, Major League Baseball etc.). Our model is more general: we do not restrict our model to deterministic settings or specific tournament formats.

We strongly believe the evaluation problem for round robin tournaments to be #P-hard though we have been unable to prove this statement. Hazon et al. showed that the problem is #P-hard for the voting system Copeland with an imperfect information model [71]. However, their results do not transfer in a straightforward manner, as their reduction requires the ability to add voters who are not candidates. Likewise Gusfield and Martel showed a similar problem of determining the probability that a team is eliminated from a round robin sports competition is #P-hard [68]. Their result does not hold in our context as their proof requires the ability to restrict the round robin schedule to a subset of the total number of games played.

Theorem 3.2.15 *Evaluation for* RR-PTBP *is in #P.*

Proof. Suppose we're given an evaluation problem with probabilities expressed with precision d. Each probability $p_{i,j}$ can be written as ${}^{n_{i,j}}/_{d}$, where $n_{i,j}$ is an integer.

We create a nondeterministic polynomial time algorithm, M, as follows:

- For each match $[i, j]$
  - M guesses $n \in [0, d)$

  - If $n < p_{i,j}$ then i wins and increments a counter for i, otherwise it increments a counter for j.

- If e^*'s counter has the highest value, accept; otherwise, reject.

- M accepts this tournament if and only if e^* wins on $\geq 1/2$ of the computations. Note that the value of 1/2 can be replaced by t with no loss of generality.

The number of accepting computations divided by the number of computations is the probability that e^* wins the tournament. ❑

The upper bound on evaluation complexity given in Theorem 3.2.15 can be extended immediately to the bribery, CCM, and exact bribery problems as well.

Theorem 3.2.16 *CCM, Constructive Bribery, and Exact Bribery for* RR-PTBP *are in* NP^{PP}.

Proof. Given a tournament T and threshold t, the $\mathrm{NP}^{\mathrm{PP}}$ machine nondeterministically guesses a set of bribes and evaluate with a single call to a #P oracle (equivalently, a PP oracle). We accept if the probability of e^* winning is $> t$. Note that we have actually shown this is in $\mathrm{NP}^{\#\mathrm{P}[1]}$. ❑

Both variants of the possible winner problem for round robin tournaments have polynomial time algorithms. While at first glance this may seem odd, it is because answering the question of possible winner does not require enumerating all possible outcomes of the tournament. Instead, to answer the possible winner problem we only need to find one configuration of the matches in which e^* wins.

The case of possible winner with access to bribes is somewhat more complicated for round robin tournaments. We begin with this problem because, as we show, the possible winner problem without access to bribes can leverage the same algorithmic construction as the possible winner with access to bribes problem. Recall the MINIMUM COST FEASIBLE FLOW problem introduced in Section 2.1.2 [1].

Name: Minimum Cost Feasible Flow

Given: A graph $G(V,E)$ with vertices V, edges E, source node s, and sink node t. Each $e \in E$ has flow, capacity, and cost. An edge labeled "$[x,y],c$" has capacity y and requires flow of at least x with cost per unit c. Cost is accumulated on a per unit flow basis (i.e., if $x = 2$ and $c = 3$ then the total cost of the edge is 6).

Question: Does there exist a flow from s to t such that the flow into each node is the same as the flow out, all minimum edge flows are satisfied, no maximum edge flows are violated, and cost is minimal?

There is a polynomial time algorithm for this problem and its solution (when the constraints are integers) is integral [1].

Using minimum cost feasible flows, Russell and Walsh provided an algorithm to compute minimal constructive manipulations (but not bribery) for deterministic round robin tournaments when the number of games that e^* can win is fixed [113]. The algorithm presented by Russell and Walsh did not allow for bribery (we do not bound the number of manipulable games) and assumed that the number of games that e^* could win is fixed. Likewise, Faliszewski et al. showed manipulation results for Copeland under deterministic information elections using feasible flows [55] and a similar construction was used by Faliszewski for the nonuniform bribery case (bribery where each voter may have different prices for changing their votes) [51]. While our construction is similar to theirs (and to an earlier constructions from Gusfield and Martel [68] and Ford and Fulkerson [64]) there are significant differences between the constructions. Our results are for a more flexible model, which is able to encapsulate bribery and manipulation, and works for even or odd numbers of entrants in the tournament.

Theorem 3.2.17 *Pos-Win-\$ for* RR-PTBP *is in P.*

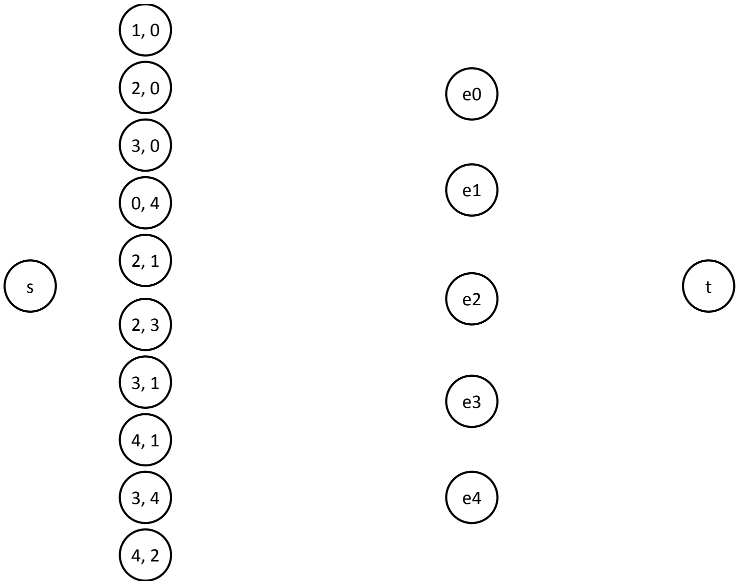
Proof. We give a polynomial time reduction of an instance of Pos-Win-$ to a minimal cost feasible flow problem. Our reduction constructs a set of minimum cost feasible flow instances; the minimum cost bribery corresponds to the cost of the flow in one of the polynomially many graphs we construct. There is a polynomial time algorithm for finding the solution to a minimum cost feasible flow problem [1]. We build this sequence of graphs so that, in each graph, if there is a way to bribe the entrants such that e^* can be made a possible winner, the cost of the bribery scheme will be equal to the cost of the minimal cost flow in that graph. We build a graph for each possible winning score e^* could have. There will be at most $n/2$ graphs built in this proof and therefore we can answer Pos-Win-$ for RR-PTBP in time polynomial in the size of the input.

We first construct the *minimum cost matrix*, $min\$_{i,j}$, as in the proof of Theorem 3.2.5. We also construct a domination graph D, which is a complete directed graph that has all entrants as nodes and each directed edge goes from a winning entrant to a losing entrant. We construct D by placing a directed edge from all entrants that are more likely to win to those less likely to win. While constructing D, we set the deterministic outcome of all games with 0 bribery cost involving e^* such that e^* is a winner (maximizing its possible wins). Once we set all the nondetermined games in favor of e^* we let $t = outDeg(e^*)$. In this instance t represents the total number of games that e^* can possibly win without making any bribes. In order for e^* to be a possible winner, we need to bribe between 0 and $n-t-1$ entrants to lose to e^* (as well as possibly influencing the outcomes of other games).

For each i from 0 to $n-t$ we construct a feasible flow graph that tells us whether e^* can be made a possible winner by bribing exactly i games involving e^*. We iterate over these graphs until we find a situation where e^* is a possible winner. Intuitively, the graph selects the cheapest bribes in order to make e^* a winner. Each unit of flow in the graph is thought of as a point for a winning entrant. The costs of edges correspond to minimum bribe costs in order to “fix” deterministic games. We arrange the flow network such that, if there is a minimum cost feasible flow, then e^* can be made a winner for cost equal to the cost of the flow.

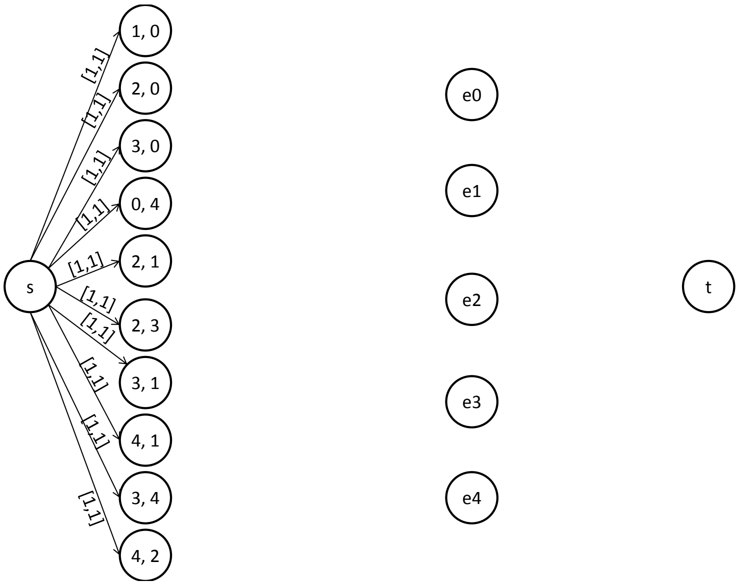
We detail the construction of the graph in three groups of edges between two interior stages. Figure 3.2.3 illustrates an example of a completely constructed graph and we proceed through a description of the construction illustrating a scenario with five entrants. The nodes other than the source and sink are classified as either left nodes or right nodes. The left half is the set of nodes that connect from the source. The right half is a set of nodes that are connected to the sink. The left and right nodes are connected by the interior edges of the graph.

Figure 3.3: Step 1 of the construction of a minimal cost winner determination graph. We have a source, sink, game nodes for each possible game, and collector nodes for each participating entrant.



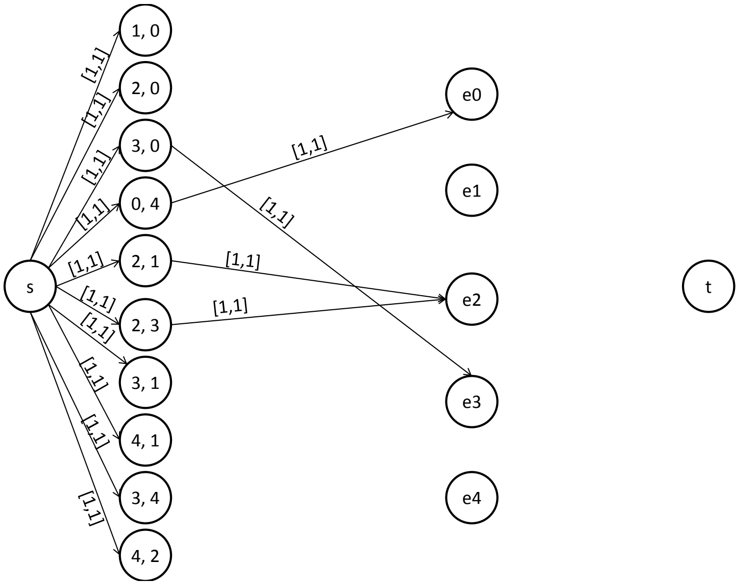
In order to build the graph we begin with a bipartite graph (not including the source and sink). The nodes on the left hand side of the graph represent all the games in the tournament. The nodes on the right hand side of the graph represent all the entrants to the tournament. Each unit of flow in this graph can be thought of, intuitively, as a “win” flowing to a particular entrant. We begin by building a source node *s*, a sink node *t*, a node for each game that will be played, and a collection node for each entrant in the tournament. The game nodes in Figure 3.2.3 are labeled with the numbers of the entrants participating in a particular game while the collector nodes are labeled with the individual entrants.

Figure 3.4: Step 2 of the construction of a minimal cost winner determination graph. We build edges from the source to all game nodes with 1 unit of flow.



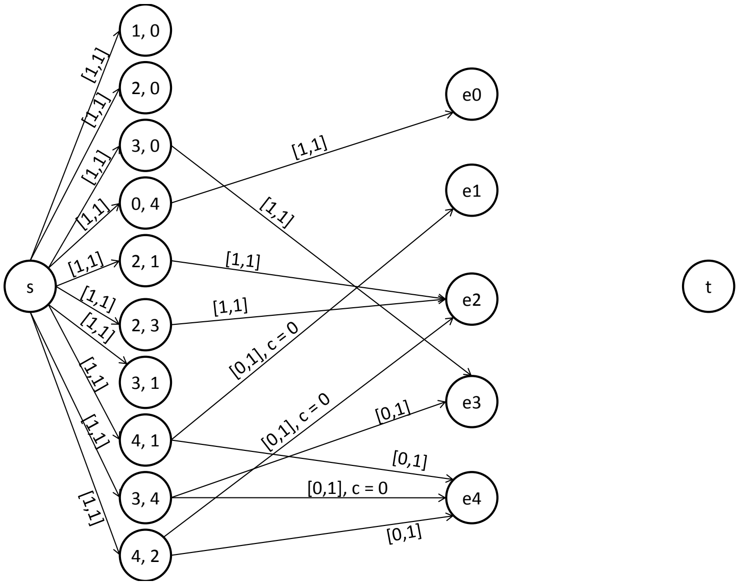
To build the edges we first set one edge from the source to each of the nodes on the left hand side with capacity and flow of 1 as illustrated in Figure 3.2.3. These units of flow will eventually count how many games a particular entrant will win. Since each game is potentially worth one point we have exactly one unit of flow into each game node.

Figure 3.5: Step 3 of the construction of a minimal cost winner determination graph. We build edges from all game nodes to their sure-to-win entrants.



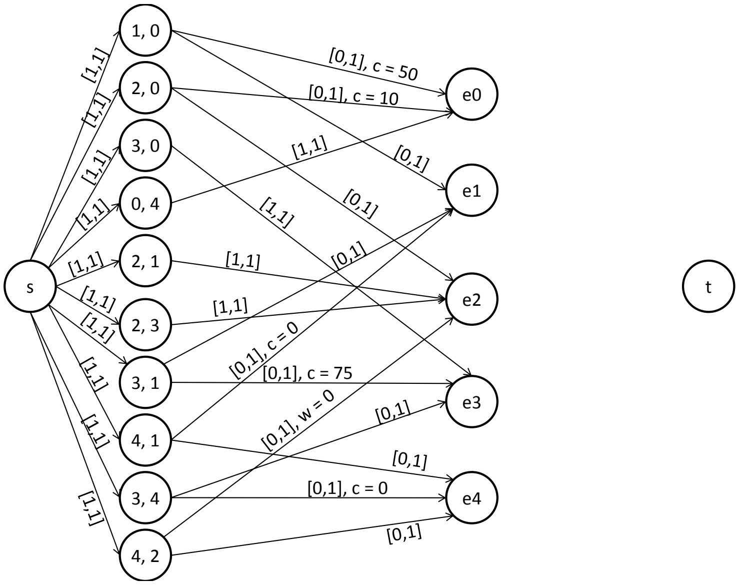
The internal edges (edges from matches to entrants) are constructed depending on the prices and probabilities of the individual games. If one entrant is sure to beat another entrant and we cannot afford any bribes with respect to these games, we build an edge $[1,1], c = 0$ from the game node to the sure-to-win entrant. Figure 3.2.3 illustrates these edges in our example graph.

Figure 3.6: Step 4 of the construction of a minimal cost winner determination graph. We build two edges in cases where either entrant is a possible winner.



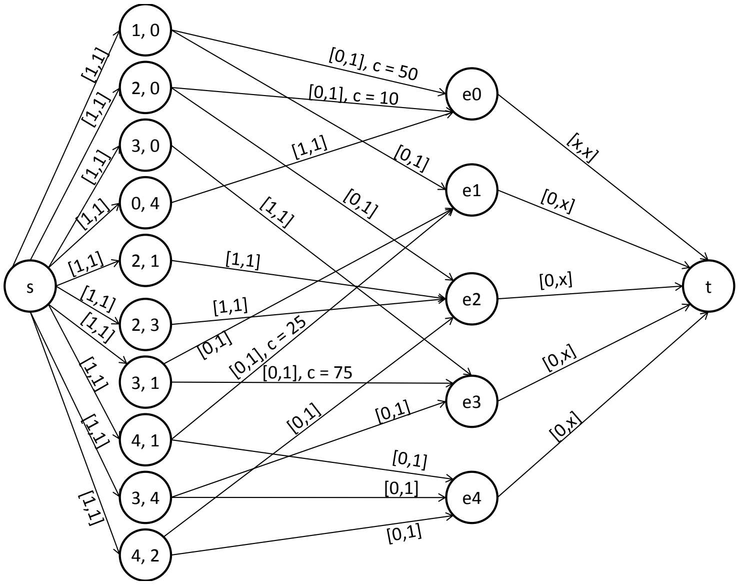
If both entrants have a nonzero probability of winning we build an edge $[0,1], c = 0$ to both entrant nodes. Since we are only attempting to find a configuration that would allow e^* to win, this construction allows us to check both possible game outcomes. This step is illustrated in Figure 3.2.3.

Figure 3.7: Step 5 of the construction of a minimal cost winner determination graph. We encode the cost of minimum bribes to change deterministic game outcomes.



The final case of internal edges is if one entrant is sure to beat another and we can possibly afford the bribe. In this case we build an edge $[0,1], c = 0$ from the game node to the sure-to-win entrant and an edge $[0,1], c = min\$_{i,j}$ from the game node to the node of the entrant who can be made a possible winner if the bribe is made. This construction allows our flow network to change the outcome of a deterministic game with cost equal to the amount of the minimum bribe. This step is illustrated in Figure 3.2.3.

Figure 3.8: A complete example of a minimal cost winner determination flow network.



We then build the final set of edges from entrant nodes to sink with capacity x, which we change between iterations of the graph. Recall that t is the total number of wins that e^* can achieve without resorting to bribery. We set $x = t + i$ and we iterate i from 0 to $n - t$ and build a graph as described above for each i. Here, x represents the most games that e^* can win making i bribes to entrants involved in games with e^*. We find the minimum cost feasible flow for each combination of bribes that e^* can make to achieve a given score. A complete graph construction is showing Figure 3.2.3.

We check each graph in turn for $i = 0, \ldots, n - t$ and if we attain a feasible flow with cost $\leq B$ we accept, otherwise, reject.

This method is complete and will find the minimum cost bribery to give e^* a chance to win. Assume that there is a cheaper flow then the one found by our network. This would mean that there existed a less expensive way to grant e^* a number of wins greater than or equal to any other entrant. Since each entrants win at most n games in any particular graph, and we check all n possible scores for e^* this cannot happen. Therefore, this construction will find the minimal cost bribery construction.

We build, at most, n networks with $\mathscr{O}(n^2)$ nodes and edges. Since the minimum cost

Table 3.4: Complexity results for deterministic tournaments. The cup and round robin results are from Russell and Walsh [113].

	Challenge	Cup	Round-Robin
Evaluation	P	P	P
CCM	P	P	P

feasible flow problem can be solved in polynomial time, the overall running time of our algorithm is polynomial. Therefore pos-win-$ for RR-PTBP is in P. ❑

We can straightforwardly extend the proof of Theorem 3.2.17 to the case where we do not have access to our budget. In the Pos-Win problem we cannot actually bribe anyone, we just have to choose winners of the non-deterministic games in a way that allows e^* a way to win. The most direct way to show this is to apply the algorithm presented in Theorem 3.2.17 with $B = 0$. We can optimize this construction somewhat but it is sufficient to state the following corollary.

Corollary 3.2.18 *Pos-Win for* RR-PTBP *is in P.*

3.2.4 Observations

Table 3.3 provides an overview of our complexity results on tournaments. Russell and Walsh [113] provided a thorough complexity analysis of similar problems in deterministic cases, all of which generate polynomial time solvable problems. Table 3.4 reveals that, in general, polynomial time evaluation procedures lead to polynomial time manipulation procedures (for deterministic systems).[6]

If we can compute a constructive manipulation in the deterministic case then we can determine a possible winner in the stochastic case. The problem of a possible winner with uncertain information is the same as constructive bribery in the deterministic setting with $B = \infty$. We don't yet have hardness proofs for some of the problems here. We believe that the problems are, in fact, complete for the classes we mention. We observe that in

[6] This is also observed in [33].

some cases the introduction of uncertainty has no effect on membership in these complexity classes while in others it represents a significant change in reasoning structure and complexity.

3.3 Summary

In this chapter we have studied the complexity of bribery and manipulation problems in elections with multiple referenda and sports tournaments, in which the actors seeking to affect the outcomes have access to only probabilities of how agents will vote or teams will perform. This chapter represents a significant departure from other work in the field of computational social choice which, generally, only considers deterministic information versions of the same problems. Many of the problems studied become computationally harder under the uncertain information setting. This increase in complexity is not uniform across problem types and, in many cases, the change in complexity seems to be closely tied to modeling choices of the particular setting.

In this chapter voters expressed their preferences over the issues and each issue was treated as an independent item. In the next chapter we investigate the situation where agents have structured preferences over the decisions to be made. This is an important step in understanding how the type and amount of information outside actors have access to affects the security of voting systems.