Nonuniform Bribery
(Short Paper)

Piotr Faliszewski
University of Rochester
Rochester, NY 14627, USA
pfali@cs.rochester.edu

ABSTRACT

We study the concept of bribery in the situation where voters are willing to change their votes as we ask them, but where their prices depend on the nature of the change we request. Our model is an extension of the one of Faliszewski et al. [9], where each voter has a single price for any change we may ask for. We show polynomial-time algorithms for our version of bribery for a broad range of voting protocols, including plurality, veto, approval, and utility-based voting. In addition we prove NP-completeness for a couple of our nonuniform bribery problems for weighted voters, and give approximation algorithms for two NP-complete bribery problems defined in [9].

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms, Theory

Keywords

preferences, computational complexity, multiagent systems

1. INTRODUCTION

Multiagent systems can often be viewed as artificial societies of autonomous agents, with each agent having his/her own set of goals, desires, and plans. Within such artificial societies, just like within natural ones, it often becomes necessary for a group of agents to arrive at a common decision (e.g., in a planning environment when no single agent can solve his/her own problems, but where they are capable of solving problems cooperatively). A very natural approach to handling such situations is to hold an election.

Unfortunately, as is well known due to theorems of Arrow, Gibbard and Satterthwaite, and Duggan and Schwartz neither there are ideal election systems nor there are ones that avoid giving agents incentive to act strategically. Bartholdi, Tovey, Trick, and Orlin [3, 2, 4], brilliantly observed that computational complexity of figuring out voter's strategic behavior might be so high, as to perhaps be enough of a barrier to prevent agents from attempting strategic actions.

Several scenarios of strategic behavior are considered in the literature. In control we assume that the organizer of the election attempts to modify their structure (e.g., via partitioning the voters into districts) in order to obtain a result most desirable for him/herself, see, e.g., [4, 10, 12, 14]. In the case of manipulation, a coalition of voters calculates what vote each member of the coalition should cast in order to obtain the result the coalition desires, see, e.g., [3, 2, 7, 5, 6, 11, 13, 14]. In bribery, see [9, 10], an external agent tries to ensure a victory of one of the candidates via bribing some voters to change their votes.

All of these problems can be studied both in the constructive case (where the goal is to ensure our favorite candidate's victory) and in the destructive case (where we try to prevent a hated candidate from winning). Also, in many situations it is natural to assume that different voters have different weights (e.g., consider the stockholders of a company or various parts of a multicriteria decision-making system that, internally, performs an election between its components, weighted based on components' confidence.)

In this paper we focus on the problem of bribery, introduced by Faliszewski et al. [9]. The authors of that paper studied bribery in several scenarios, depending on the voting system used and whether the voters were weighted and/or were assigned price-tags for changing their votes. In particular, in the priced cases they assumed that each voter is willing to change his/her vote arbitrarily, provided that the briber pays the fixed-per-voter price. This assumption is fairly unrealistic. For example, consider an election with three candidates, a, b, and c, and a particular voter who prefers a to b to c, but who actually likes both a and b and who absolutely hates c. Such a voter may be willing, at a small price, to change his/her vote to rank b first, but would never, regardless of the bribe, change the vote to rank c first.

A different example where it is useful to model voters as having such nonuniform prices is best seen from the point of view of the briber. A briber that wants some candidate p to win, might want to follow a certain *policy* in his/her bribing. For example, he/she might not want to bribe anyone to vote for p in order not to cast "bad light" on p. Such a briber would have to make p a winner via bribes that redistribute other candidate's support. Using the nonuniform model of bribery one could express this policy via setting the prices for voting for p so high as to be outside of the allowed budget. This policy was studied in [9] but, other interesting policies exist. e.g., limiting briber's ability to change votes.

Cite as: Nonuniform Bribery (Short Paper), P. Faliszewski, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May,12-16.,2008,Estoril,Portugal,pp. 1569-1572.

Yet another scenario where nonuniform bribery model is useful regards the issue of coalition formation. Consider an election where one of the voters realizes that his/her option is very unlikely to win, but where there are many agents (voters) that support options similar, but slightly different. Such an agent might want to find out which of the others, but as few as possible, he/she would have to convince to form a coalition with him/her in order to have enough voting power as to choose an option that all of them would be reasonably satisfied with. One way to compute this set would be to: (a) find a group of voters that currently vote for options similar to the agent's, (b) form nonuniform bribery instance where those voters can be bribed at a relatively low price to vote for the agent's option and all other briberies are either very expensive or impossible (beyond budget), (c) compute a minimum-cost nonuniform bribery that ensures that the agent's favorite option wins. The voters involved in this bribery would be candidates for the coalition.

Thus, we believe that the issue of nonuniform bribery is both important and useful, even in the cases where we are not really "bribing" anyone, but simply are trying to strategically plan our behavior. In this paper we give a number of results that show that the problem of nonuniform bribery can often be solved in polynomial time in the case of unweighted voters, yet becomes NP-complete if the voters are weighted. We also give several approximation algorithms for previously studied NP-complete bribery problems.

2. PRELIMINARIES

We view an election as a pair (C, V), where C is a set of candidates, $C = \{c_1, \ldots, c_m\}$, and V is a multiset of voters, each represented via his/her preference over C. In this paper we represent each voter's preference as an m-dimensional vector of nonnegative integers, indicating voter's perceived utility from electing each candidate (such utility-based voting was studied in the context of election manipulation, e.g., by Elkind and Lipmaa [8]).

Definition 2.1. *Let k and b be two positive integers. By a (k, b)-election we mean an election over some candidate set $C = \{c_1, \ldots, c_m\}$, where each voter from the voter set V distributes k integral points among the candidates, never assigning more than b points to a single candidate. In the end, the candidates with most points are the winners.*

A free-form (k, b)-election is a (k, b)-election where voters can choose not to use all of their points.[1]

Note that if we, e.g., have to few candidates or k is too large then a (non free-form) (k, b)-election is impossible.

The standard preference model studied in computational social choice literature assumes that each voter has a strict linear order of preference over all candidates. Our model is more appropriate for utility-based voting, and is powerful enough to capture such voting rules as plurality (as $(1, 1)$-elections; in plurality elections each voter assigns one point to his/her favorite candidate), veto (as $(m - 1, 1)$-elections; in veto election each voter gives a single point to everyone except for his/her most hated option), approval (as free-form $(m, 1)$-elections; in approval each voter either approves or disapproves of each of the candidates and the candidates with most approvals win), and t-approval (as $(t, 1)$-elections; as approval, but each candidate has to approve of exactly t candidates). Utility-based voting is a very attractive concept and we believe it is interesting to study it in computational context. In the next section we will see a natural way of expressing our idea of nonuniform bribery within (k, b)-elections via introducing, possibly distinct, prices for moving points between candidates. Similar schemes that we came up with for preference-order-based voting were neither as elegant nor easy to work with.

We now provide notation for our main algorithmic tool, flow networks. A flow network is defined via a set of nodes $N = \{s, t, n_1, \ldots, n_m\}$, where s is the source and t is the sink, a capacity function $cpc : N \times N \to \mathbb{N}$, and a cost function $prc : N \times N \to \mathbb{N}$. We say that two nodes, call them u and v, are connected if $cpc(u, v) > 0$. Note that $cpc(u, v)$ does not need to equal $cpc(v, u)$; the channels connecting two nodes are unidirectional. A function $f : N \times N \to \mathbb{Z}$ is a flow in such a network if it satisfies the following constraints: (a) $(\forall_{u,v \in N})[f(u, v) \leq cpc(u, v)]$ (i.e., we do not send flow beyond capacity), (b) $(\forall_{u,v \in N})[f(u, v) = -f(v, u)]$, and (c) $(\forall_{u \in N - \{s,t\}})[\sum_{v \in N - \{u\}} f(u, v) = 0]$. We interpret $f(u, v) = t$, $t > 0$, as t units of flow traveling directly from u to v. A negative flow value indicates reversed direction of travel. The value of a flow is defined as $\sum_{u \in N - \{s\}} f(s, u)$ and its cost is $\sum_{u,v \in N \mid f(u,v) \geq 0} prc(u, v) f(u, v)$. I.e., for each two nodes u, v we pay $prc(u, v)$ for sending each single unit of flow from u to v.

In the min-cost flow problem we are given a nonnegative integer K, a set of nodes with source and sink, a capacity function, and a cost function, and our goal is to find a minimum-cost flow of value K (or to indicate that such a flow doesn't exist). Min-cost flow is solvable in polynomial time, see, e.g., an excellent monograph of Ahuja et al. [1]. (We note that Procaccia et al. [15] also use flow networks in the context of achieving proportional representation.)

An algorithm A is a fully polynomial-time approximation scheme (FPTAS) for a given minimization problem (e.g., a problem of finding minimum-cost bribery) if (a) algorithm A on input (I, ε), where I is an instance of the problem and ε is a positive real number, $0 < \varepsilon < 1$, runs in time polynomial in the size of I and $\frac{1}{\varepsilon}$, and (b) A has the property that if O is the value of the optimal solution for I then $A(I, \varepsilon)$ produces a solution with value S such that $S \leq (1 + \varepsilon)O$. We stress that algorithm A has to output a correct solution for the problem, only that the cost of this solution may be somewhat above the optimum.

3. NONUNIFORM BRIBERY

Let k and b be two positive integer values (possibly dependent on the number of candidates and voters in E; see the next sentence). We define (k, b)-bribery problem as follows: The input is a (k, b)-election E with candidate set $C = \{c_1, \ldots, c_m\}$ and a multiset V of voters $v_1, \ldots, v_n$, where each voter v_i is represented via an m-dimensional integer vector describing in an obvious way how many points v_i assigns to which candidates, a nonnegative integer B (the budget), and for each voter v_i a price function $\pi_i : C \times C \to \mathbb{N}$. A unit bribery involves asking some voter $v_\ell \in V$ to move a single point that v_ℓ currently assigns to some candidate c_i to another candidate c_j. The cost of such a unit bribery is $\pi_\ell(c_i, c_j)$. (Naturally, for each $\ell \in \{1, \ldots, n\}$ and

[1]Note: A (k, b)-election might be impossible, e.g., if k is too large or if we have too few candidates. On the other hand, free-form (k, b)-election is always possible.

each candidate c_i we have $\pi_\ell(c_i, c_i) = 0$.) The question is if given the budget B, this (k, b)-election E, and functions $\pi_1, \dots, \pi_n$ (one for each voter), it is possible to perform a set of unit briberies of total cost at most B, such that (a) preferred candidate $p = c_1$ becomes a winner, and, (b) each voter assigns at most b points to each candidate (i.e., the rules of a (k, b)-election are not broken). We explicitly require that all the unit briberies are executed "in parallel," that is, a briber cannot first bribe voter v_ℓ to move a point from some candidate c_i to another candidate c_j and afterward move that same point from c_j to yet another candidate c_q.[2] (Though, it would still be legal to move to c_q a point that v_ℓ had assigned to c_j before the bribery. Of course, points are anonymous, unnamed, entities; our requirement of not moving "the same" point twice formally means that briberies are only legal if for each voter v_ℓ they move, within the preference vector of that voter, at most as many points away from each candidate as many that candidate had been assigned by v_ℓ before bribery.)

Free-form (k, b)-bribery is defined analogously, only that voters do not have to assign all their points to candidates and, in addition to other unit briberies, we can bribe voters to either use their unassigned points or to remove some of their points from particular candidates. In order to accomplish this we extend our price functions to incorporate null candidate ε representing the slot for unassigned points (naturally, the b-bound does not apply to ε).

We now give our main result regarding (k, b)-bribery.

THEOREM 3.1. *There is an algorithm that solves both (k, b)-bribery instances and free-form (k, b)-bribery instances in time polynomial in k* and *the size of the instance.*

PROOF. Due to length restrictions we only sketch the proof for (k, b)-bribery, skipping the free-form variant of the problem. Our input is a (k, b)-election E, with candidate set $C = \{c_1, \dots, c_m\}$ and voter multiset $V = \{v_1, \dots, v_n\}$, a nonnegative integer B (the budget), and voters' price functions $\pi_1, \dots, \pi_n$. Our goal is to ensure that candidate $p = c_1$ is a winner of the election via a bribery of cost at most B.

Our proof follows via constructing a series of flow networks and computing min-cost solutions for them. The intuition here is that the points that voters assign to candidates are modeled via the units of flow traveling in the network. We design our networks in such a way that minimizing the cost of the flow, in essence, maximizes the number of points our designated candidate $p = c_1$ via a minimum-cost bribery.

We know that each candidate can at most receive kn points. For each nonnegative integer K between 1 and kn our algorithm tests if there is a bribery of cost at most B that ensures that p receives exactly K points and every other candidate receives at most K points. Let us now fix a value of K and show how such a test can be executed.

We form a network flow with node set $N = \{s, t\} \cup \bigcup_{i=1}^{n} C_i \cup \bigcup_{i=1}^{n} C'_i \cup F$, where $F = \{f_1, \dots, f_m\}$ and for each $i \in \{1, \dots, n\}$ we have $C_i = \{c_{i1}, \dots, c_{im}\}$, $C'_i = \{c'_{i1}, \dots, c'_{im}\}$. For each $i \in \{1, \dots, n\}$, nodes in the sets C_i represent point distribution of voter v_i before bribery, nodes in C'_i represent point distribution of voter v_i after the bribery, and nodes in F are used to enforce the rules of (k, b)-election and to sum points for all candidates.

[2]All our results stay if such sequential bribing was legal. We believe that "parallel bribery mode" is more appropriate.

We introduce the following capacities and costs for edges in our network. (All unmentioned edges have capacity 0.) For each voter v_ℓ and candidate c_i we have $cpc(s, c_{\ell i})$ equal to the number of points v_ℓ assigns to c_i before bribery and $prc(s, c_{\ell i}) = 0$. These edges model delivering appropriate number of points to nodes from sets $C_1, \dots, C_n$.

For each node $c_{\ell i}$ and each candidate c_j we have $cpc(c_{\ell i}, c'_{\ell j}) = k$ and $prc(c_{\ell i}, c_{\ell j}) = \pi_\ell(i, j)$. These edges model unit briberies. The briber can ask each voter to move his points as the briber likes, but pays appropriate price for moving each point.

For each node $c'_{\ell i}$ we set $cpc(c'_{\ell i}, f_i) = b$ and $prc(c'_{\ell, f_i}) = 0$. These edges enforce that no candidate can receive more than b points from a single voter. Finally, for each node f_i, we have $cpc(f_i, t) = K$, $prc(f_1, t) = 0$, and for all $i \in \{2, \dots, m\}$ we have $prc(f_i, t) = T$, where T is an integer higher than the cost of any possible bribery (e.g., take $T = 1 + kn \max_{\ell,i,j} \pi_\ell(c_i, c_j)$).

To perform our test we compute minimum-cost flow of value kn in this network. If such a flow doesn't exist then it means that there is no legal way of distributing points via bribery in such a way that each voter has score at most K. In such a case we disregard this value of K and continue with the next one. We can interpret our minimum-cost flow as follows: Each set of nodes $C_1, \dots, C_n$ receives units of flow corresponding to the distribution of points of each voter (because flow has value kn and because of the capacities of edges connecting the source with nodes in $C_1, \dots, C_n$) and units of flow travel from nodes in sets C_i to nodes in sets C'_i, respectively, modeling our bribery. Finally, they all accumulate in nodes from set F, from where they all reach the sink. The nature of edges between nodes in C'_i's and nodes in F guarantees that we arrive with a legal distribution of points for each voter. The cost of this flow can be expressed as $T \cdot (kn - p\text{'s score after bribery}) + \text{cost-of-bribery}$. Since T is chosen to be larger than any possible cost of bribery, we know that minimum cost enforces that node f_1, corresponding to $p = c_1$, receives as many units of flow as legally possible. If p can legally receive K units of flow then minimum-cost flow delivers this many units; the capacity of the edge linking $p = c_1$ with the sink is K so the flow cannot deliver more. If the flow cannot legally deliver K units of flow to c_1 then we can safely disregard this network (because we have already handled this flow when analyzing smaller values of K). Interpretation of our flow gives that our bribery guarantees that p gets exactly K points and all the other candidates receive at most K points each (as each node $f_2, \dots, f_n$ can only deliver K units of flow to the sink). This is achieved via a minimum cost bribery as the cost-of-bribery is the remaining part of the cost of our flow.

This way we test in polynomial-time for each K if there is a nonuniform bribery of cost at most B that ensures that p receives exactly K points and all other candidates receive at most K points. In total, the running time of our algorithm is polynomial in the size of our election and k. $\square$

COROLLARY 3.2. *Nonuniform bribery is solvable in polynomial time for: plurality, veto, (t-)approval, and utility-based voting where the number of points each voter can distribute is polynomial in the number of candidates.*

All the rules mentioned above can be expressed as appropriate (k, b)-elections and our main result applies. Naturally, we ask if the result also holds for weighted voters? Unfortunately, in general the answer is "no." (k, b)-weighted-bribery is an analog of (k, b)-bribery, with weighted voters

Theorem 3.3. *$(1, 1)$-weighted-bribery is* NP*-complete.*

The proof, skipped for space reasons, follows via a reduction from plurality-weighted-negative-bribery, defined in [9]. $(m - 1, 1)$-weighted-bribery, where m is the number of candidates, is NP-complete as well via a reduction from veto-weighted-bribery, a problem studied in [9] as well.

We cannot hope for a general, efficient, algorithm handling (k, b)-weighted-bribery for k, b polynomially bounded in the input size, but there are interesting special cases. We focus on the restriction of $(1, 1)$-weighted-bribery to the case where each voter has a single price for moving his/her point to any of the candidates. This problem is called plurality-weighted-\$bribery in [9] and we will use this name. We also consider a restriction of free-form $(m, 1)$-weighted-bribery, where m is the number of candidates, to a situation where each voter has for each candidate a price of flipping the support for that candidate. This restriction is equivalent to what in [9] is called approval-weighted-\$bribery′. These two problems can be solved approximately.

Theorem 3.4. *plurality-weighted-\$bribery and approval-weighted-\$bribery′ both have fully polynomial-time approximation schemes (FPTAS).*

We skip the details of the proof, but we outline the main idea. For both of our problems, Faliszewski et al. [9] gave algorithms that work in time polynomial in the size of the instance *and* the value of the highest prime. The idea of our FPTAS is to divide the prices by an appropriately large factor so that the largest price has value at most polynomial in the size of a given instance. However, simply taking this approach is not enough as the largest price might be enormous and yet not used in the optimal solution. Scaling the prices taking into account very large price range may essentially remove all the information about the prices that actually participate in the optimal solution. Our FPTAS performs polynomially many "scalings" of prices, starting with a very small scaling factor and up to one appropriate for the largest occurring price. Small factors do not reduce large prices enough, so—when using a small factor—we remove prices that are too large and replace them with appropriately smaller ones, that nonetheless are too large to be used in a successful bribery.

It is unlikely that a general FPTAS for (k, b)-weighted-bribery exists. If we had one for $(1, 1)$-weighted-bribery then via a good enough approximation, one could solve plurality-weighted-negative-bribery. This would imply P = NP.

4. FUTURE WORK

There are several interesting open directions of study. In particular, it would be interesting to study notions similar to our nonuniform bribery, but in the context of preferences represented via linear orders. We are also interested in further results regarding approximation of bribery and other election-related problems.

5. ACKNOWLEDGEMENTS

This work was supported in part by grant NSF-CCF-0426761. The author would like to thank Edith Hemaspaandra, Lane Hemaspaandra, Jörg Rothe and Henning Schnoor for interesting discussions and the anonymous referees for their helpful comments.

6. REFERENCES

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.

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

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

[4] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[5] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. of the 18th Int. Joint Conf. on A. I.*, pages 781–788. Morgan Kaufmann, August 2003.

[6] V. Conitzer and T. Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proc. of the 21st National Conf. on Art. Int.*, pages 627–634. AAAI Press, July/August 2006.

[7] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):1–33, 2007.

[8] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *The 16th Annual International Symposium on Algorithms and Computation, ISAAC 2005*, pages 206–215. Springer-Verlag *Lecture Notes in Computer Science #3872*, December 2005.

[9] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. The complexity of bribery in elections. In *Proc. of the 21st National Conf. on Art. Int.*, pages 641–646. AAAI Press, July 2006.

[10] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting broadly resist bribery and control. In *Proc. of the 22nd National Conf. on Art. Int.*, pages 724–730. AAAI Press, July 2007.

[11] E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

[12] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5-6):255–285, April 2007.

[13] A. Procaccia and J. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, February 2007.

[14] A. Procaccia, J. Rosenschein, and A. Zohar. Multi-winner elections: Complexity of manipulation, control, and winner-determination. In *Proc. of the 20th Int. Joint Conf. on A. I.*, pages 1476–1481. AAAI Press, January 2007.

[15] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 2007.