International Game Theory Review, Vol. 13, No. 1 (2011) 1–32

DOI: 10.1142/S0219198911002824

COST AND COMPLEXITY OF HARNESSING GAMES WITH PAYMENTS

RAPHAEL EIDENBENZ[*,§], YVONNE ANNE PIGNOLET[†,¶], STEFAN SCHMID[‡,‖] and ROGER WATTENHOFER[*,**]

[*] *Computer Engineering and Networks Laboratory (TIK)*
ETH Zurich, Switzerland

[†] *IBM Research, Zurich Laboratory*
Switzerland

[‡] *Deutsche Telekom Laboratories/TU Berlin*
Berlin, Germany

[§] *eidenbenz@tik.ee.ethz.ch*
[¶] *yvo@zurich.ibm.com*
[‖] *stefan@net.t-labs.tu-berlin.de*
[**] *wattenhofer@tik.ee.ethz.ch*

This article studies how a mechanism designer can influence games by promising payments to the players depending on their mutual choice of strategies. First, we investigate the cost of implementing a desirable behavior and present algorithms to compute this cost. Whereas a mechanism designer can decide efficiently whether strategy profiles can be implemented at no cost at all our complexity analysis indicates that computing an optimal implementation is generally **NP**-hard. Second, we introduce and analyze the concept of *leverage* in a game. The leverage captures the benefits that a benevolent or a malicious mechanism designer can achieve by implementing a certain strategy profile region within economic reason, i.e., by taking the implementation cost into account. Mechanism designers can often manipulate games and change the social welfare by a larger extent than the amount of money invested. Unfortunately, computing the leverage turns out to be intractable as well in the general case.

1. Introduction

Many societies and distributed systems exhibit a socio-economic complexity that is often difficult to describe and understand formally from a scientific perspective. Game theory is a powerful tool for analyzing decision making in systems with autonomous and rational (or selfish) participants. It is used in a wide variety of fields such as biology, economics, politics, or computer science. A major achievement of game theory is the insight that networks of self-interested agents (or *players*) often suffer from inefficiency due to effects of selfishness. The concept of the price of anarchy allows us to quantify these effects: The price of anarchy compares the performance of a distributed system consisting of selfish participants to the performance of an optimal reference system where all participants collaborate perfectly.

If a game theoretic analysis of a distributed computing system reveals that the system has a large price of anarchy, this indicates that the protocol should be extended by a mechanism encouraging cooperation. In many distributed systems, for example in a computer network, a mechanism designer cannot change the rules of interactions. However, she may be able to influence the players' behavior by offering payments for certain outcomes. On this account, we consider a mechanism designer whose power is to some extent based on her monetary assets, primarily, though, on her *creditability*. That is, the players trust her to pay the promised payments. Thus, a certain subset of outcomes is implemented in a given game if, by expecting additional non-negative payments, rational players will necessarily choose one of the desired outcomes. A designer faces the following optimization problem: How can the desired outcome be implemented at minimal cost? Surprisingly, it is sometimes possible to improve (or worsen) the performance of a given system merely by creditability, i.e., without any payments at all: promising payments for other profiles can function as some sort of insurance upon which players choose a better strategy, ending up in a profile where eventually no payments are made.

Whether a mechanism designer is willing to invest the cost of implementing a desired outcome often depends on how much better than the original outcome the implemented outcome is. If the social welfare gain does not exceed the implementation cost, the mechanism designer might decide not to influence the game at all. In many games, however, manipulating the players' utility is profitable. The following extension of the well-known prisoners' dilemma illustrates this phenomenon. Two bank robbers, both members of the *Al Capone gang*, are arrested by the police. The policemen have insufficient evidence for convicting them of robbing a bank, but they could charge them with a minor crime. Cleverly, the policemen interrogate each suspect separately and offer both of them the same deal. If one testifies to the fact that his accomplice has participated in the bank robbery, they do not charge him for the minor crime. If one robber testifies and the other remains silent, the former goes free and the latter receives a three-year sentence for robbing the bank and a one-year sentence for committing the minor crime. If both betray the other, each of them will get three years for the bank robbery. If both remain silent, the police can convict them for the minor crime only and they get one year each. There is another option, of course, namely to confess to the bank robbery and thus supply the police with evidence to convict both criminals for a four-year sentence (cf. G in Fig. 1; note that payoffs are expressed in terms of *saved* years!). A short game-theoretic analysis shows that a player's best strategy is to testify. Thus, the prisoners will betray each other and both get charged a three-year sentence. Now assume that Mr. Capone gets a chance to take influence on his employees' decisions. Before they take their decision, Mr. Capone calls each of them and promises that if they both remain silent, they will receive money compensating for one year in jail (for this scenario, we presume that time really is money!) and furthermore,

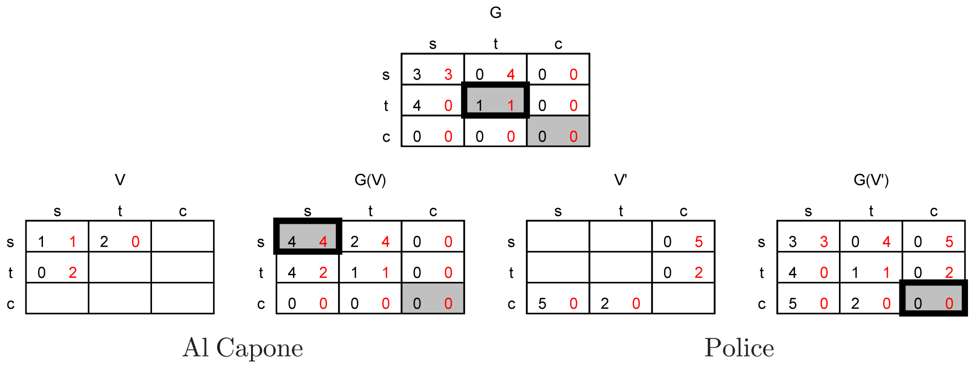

Fig. 1. Extended prisoners' dilemma: G shows the prisoners' initial payoffs, where payoff values equal *saved years*. The first strategy is to remain silent (s), the second to testify (t) and the third to confess (c). Nash equilibria are colored gray, and non-dominated strategy profiles have a bold border. The left bimatrix V shows Mr. Capone's offered payments which modify G to the game $G(V)$. By offering payments V', the police implements the strategy profile (c, c). As $V_1(c, c) = V_2(c, c) = 0$, payments V' implement (c, c) for free.

if one remains silent and the other betrays him, Mr. Capone will pay the former money worth two years in prison (cf. V in Fig. 1). Thus, Mr. Capone creates a new situation for the two criminals where remaining silent is the most rational behavior. Mr. Capone has saved his gang an accumulated two years in jail.

Let us consider a slightly different scenario where after the police officers have made their offer to the prisoners, their commander-in-chief devises an even more promising plan. He offers each criminal to drop two years of the four-year sentence in case he confesses the bank robbery and his accomplice betrays him. Moreover, if he confesses and the accomplice remains silent they would let him go free and even reward his honesty with a share of the booty (worth going to prison for one year). However, if both suspects confess the robbery, they will spend four years in jail. In this new situation, it is most rational for a prisoner to confess. Consequently, the commander-in-chief implements the best outcome from his point of view without dropping any sentence and he increases the accumulated years in prison by two.

From Mr. Capone's point of view, implementing the outcome where both prisoners keep quiet results in four saved years for the robbers. By subtracting the implementation cost, the equivalent to two years in prison, from the saved years, we see that this implementation yields a benefit of two years for the Capone gang. We say that the *leverage* of the strategy profile where both prisoners play s is two. For the police, the leverage of the strategy profile where both prisoners play c is two, since the implementation costs nothing and increases the years in prison by two. Since implementing c reduces the players' gain, we say the strategy profile where both play c has a *malicious leverage* of two.

In the described scenario, Mr. Capone and the commander-in-chief solve the optimization problem of finding the game's strategy profile(s) which bear the largest (malicious) leverage and therewith the problem of implementing the corresponding outcome at optimal cost.

In the remainder of this section, we review related work and give an overview of our contributions, followed by an introduction of our model and some basic game theoretic definitions.

1.1. *Related work and our contributions*

Game theory (e.g., Osborne and Rubinstein [1994]) and mechanism design & implementation theory (Maskin [1999]; Maskin and Sjöström [2002]) have been a flourishing research field for many decades. In 2007, three pioneers in implementation theory (Leonid Hurwicz, Eric Maskin, and Roger Myerson) were awarded the Nobel prize. With the advent of the Internet and its numerous applications such as e-commerce (e.g., Feigenbaum and Shenker [2003]; Rosenschein and Zlotkin [1994]), peer-to-peer systems (e.g., Dash *et al.* [2003]), or social networks, algorithmic mechanism design and game theory is extensively studied by computer scientists as well. For instance, game theory is used to shed light onto sociological and economic phenomena in decentralized networks consisting of different interacting stake-holders, and mechanism design is needed to ensure efficiency in online auctions like eBay. For an interesting recent survey of the field, we refer the reader to the book by Nisan *et al.* [2007].

Popular problems in computer science studied from a game theoretic point of view include *virus propagation* (Aspnes *et al.* [2005]), *congestion* (Christodoulou and Koutsoupias [2005]), *wireless spectrum auctions* (Zhou *et al.* [2008]), among many others. Poor performance of selfish networks requires research for countermeasures (Dash *et al.* [2003]; Maskin and Sjöström [2002]). Cole *et al.* [2003a,b] have studied how incentive mechanisms can influence selfish behavior in a routing system where the latency experienced by the network traffic on an edge of the network is a function of the edge congestion, and where the network users are assumed to selfishly route traffic on minimum-latency paths. They show that by pricing network edges the inefficiency of selfish routing can always be eradicated, even for heterogeneous traffic in single-commodity networks.

We believe that the model studied in this article is particularly interesting for computer networks. Computer networks have special boundary conditions that preclude certain classic implementation theoretic solutions. For example, it is difficult for a mechanism designer to influence the rules according to which the players act, e.g., by laws. One way of manipulating the players' decision-making is to offer them money for certain outcomes. Monderer and Tennenholtz [2003] study a minimal rationality model (players choose non-dominated strategies) and show how creditability can be used to outwit selfish agents and influence their decisions. They consider a mechanism designer who cannot enforce behaviors and cannot change the system, and who attempts to encourage agents to adopt desired behaviors in a given multi-player setting. The only way the third party can influence the course of the game is by promising non-negative monetary transfers for certain outcomes (notion of *k-implementation*). The interested party wishes to minimize her expenses to implement certain outcomes. The authors show that the mechanism designer might be able to induce a desired outcome at very low cost. In particular, they prove that any pure Nash equilibrium has a *0-implementation* (see also Dybvig and Spatt [1983]; Segal [1999]; Spiegler [2000]), i.e., it can be transformed into a dominant strategy profile at zero cost (achieving a Price of Stability for free, e.g., Resnick *et al.* [2009]). Similar results hold for any given ex-post equilibrium of a frugal VCG mechanism. Moreover, the paper addresses the question of the hardness of computing the minimal implementation cost.

We extend Monderer and Tennenholtz [2003] in various respects. Monderer and Tennenholtz [2003] attends to mechanism designers calculating with maximum possible payments for a desired outcome — a "worst-case scenario". To assume the worst case makes sense since it is left open how a player chooses among the non-dominated strategies. In this article we also consider games where, due to the lack of information of other players' payoff functions, a player is assumed to pick a one of her non-dominated strategies uniformly at random. For such a manner of dealing with imperfect knowledge or uncertainty, we prove that computing the optimal implementation cost is **NP**-hard in general. Analyzing the computational complexity of worst-case scenarios turns out to be more intricate; we discovered an error in the approach taken in Monderer and Tennenholtz [2003], and it is unclear how to repair their construction.

We introduce the concept of leverage, a measure for the change of behavior a mechanism design can inflict, taking into account the social gain and the implementation cost. Regarding the payments offered by the mechanism designer as some form of insurance, it seems natural that outcomes of a game can be improved at no cost. On the other hand, we show that a malicious mechanism designer can in some cases even reduce the social welfare at no cost. Second, we present algorithms to compute both the beneficial as well as the malicious leverage, and provide evidence that several optimization problems related to the leverage are **NP**-hard.

To the best of our knowledge, this is the first work studying malicious mechanism designers which aim at influencing a game based primarily on their creditability. Other types of maliciousness have been studied before in various contexts, especially in cryptography, and it is impossible to provide a complete overview of this literature. Recently, the concept of *BAR games* (Aiyer *et al.* [2005]) has been introduced which aims at understanding the impact of altruistic and malicious behavior in game theory. Moscibroda *et al.* [2006] extend the virus inoculation game from Aspnes *et al.* [2005] to comprise both selfish and malicious players. A similar model has recently been studied in the context of congestion games (Babaioff *et al.* [2007]). Our work is also related to *Stackelberg theory* Roughgarden [2001] where a fraction of the entire population is orchestrated by a global leader. In contrast to our model, the leader is not bound to offer any incentives to follow her objectives.

In the recent research thread of *combinatorial agencies* (Babaioff *et al.* [2006a,b]; Eidenbenz and Schmid [2009]), a setting is studied where a mechanism designer seeks to influence the outcome of a game by contracting the players individually; however, as she is not able to observe the players' actions, the contracts can only depend on the overall outcome.

Our work has also connections to fault-tolerant mechanism design: In Porter *et al.* [2008], the authors extend the field of mechanism design to take into account execution uncertainty, where the costs of a player depends on the probabilities of failure. Apart from incentive-compatible mechanisms, they also give impossibility results. Moreover there are intriguing touching points with correlated equilibria and mediated mechanisms, where a mechanism designer can communicate with the players and suggest (without money) certain subset of the outcomes for example (e.g., Monderer and Tennenholtz [2009]); indeed, in Monderer and Tennenholtz [2003] it is shown that all correlated equilibria can in fact be 0-implemented. Recently, Bradonjic *et al.* [2009] also introduced the study of a malicious interested party in the mediator setting.

Preliminary versions of this work have been published at the International Conference on Combinatorial Optimization and Applications (Eidenbenz *et al.* [2007b]) and the International Symposium on Algorithms and Computation (Eidenbenz *et al.* [2007a]). Follow-up work by Moscibroda and Schmid [2009] studies an application of the theories devised in this article to the domain of throughput maximization.

1.2. *Preliminaries and model*

1.2.1. *Game theory*

A finite *strategic game* can be described by a tuple $G = (N, X, U)$, where $N = \{1, 2, \ldots, n\}$ is the set of *players* and each player $i \in N$ can choose a *strategy* (action) from the set X_i. The product of all the individual players' strategies is denoted by $X := X_1 \times X_2 \times \cdots \times X_n$. In the following, a particular outcome $x \in X$ is called *strategy profile* and we refer to the set of all other players' strategies of a given player i by $X_{-i} = X_1 \times \cdots \times X_{i-1} \times X_{i+1} \times \cdots \times X_n$. An element of X_i is denoted by x_i, and similarly, $x_{-i} \in X_{-i}$; we may write x_i, x_{-i} to denote strategy profile $x \in X$ where player i plays x_i and all other players play according to x_{-i}. Finally, $U = (U_1, U_2, \ldots, U_n)$ is an n-tuple of *payoff functions* (utilities), where $U_i : X \to \mathbb{R}$ determines player i's payoff arising from the game's outcome. The *social gain* of a game's outcome is given by the sum of the individual players' payoffs at the corresponding strategy profile x, i.e. $gain(x) := \sum_{i=1}^{n} U_i(x)$. Let $x_i, x'_i \in X_i$ be two strategies available to Player i. We say that x_i *dominates* x'_i iff $U_i(x_i, x_{-i}) \geq U_i(x'_i, x_{-i})$ for every $x_{-i} \in X_{-i}$ and there exists at least one x_{-i} for which a strict inequality holds. x_i is the *dominant* strategy for player i if it dominates every other strategy $x'_i \in X_i \backslash \{x_i\}$. x_i is a *non-dominated* strategy if no other strategy dominates it. By $X^* = X_1^* \times \cdots \times X_n^*$ we will denote the set of non-dominated strategy profiles, where X_i^* is the set of non-dominated strategies available to the individual player i. A *strategy profile set* — also called *strategy profile region* — $O \subseteq X$ of G is a subset of all strategy profiles X, i.e., a region in the payoff matrix consisting of one or multiple strategy profiles. Similarly to X_i and X_{-i}, we define $O_i := \{x_i | \exists\, x_{-i} \in X_{-i}$ s.t. $(x_i, x_{-i}) \in O\}$ and $O_{-i} := \{x_{-i} | \exists\, x_i \in X_i$ s.t. $(x_i, x_{-i}) \in O\}$.

1.2.2. *Implementation cost*

Our model is based on the classic assumption that players are rational and always choose a non-dominated strategy. Additionally, we assume that players do not collude. We examine the impact of payments to players offered by a *reliable mechanism designer* (an interested third party) who seeks to influence the outcome of a game. It is assumed that the mechanism designer has complete knowledge of the players' utilities. By *reliable* we mean that the owed payments will always be acquitted. Note that this differs from standard mechanism design where a designer (e.g., a government) defines an interaction for self-motivated parties that will allow it to obtain some desired goal (such as maximizing revenue or social welfare) taking the agents' incentives into account, see also the discussion in Monderer and Tennenholtz [2003]. In many distributed systems, unfortunately, interested parties cannot control the rules of interactions. A network manager for example cannot simply change the communication protocols in a given distributed systems in order to lead to desired behaviors, and a broker cannot change the rules in which goods are sold by an agency auctioneer to the public.

The payments promised by the mechanism designer are described by a tuple of non-negative payment functions $V = (V_1, V_2, \ldots, V_n)$, where $V_i : X \to \mathbb{R}^+$, i.e., the payments for player i depend on the strategy Player i selects as well as on the choices of all other players. Thereby, we assume that the players trust the mechanism designer to finally pay the promised amount of money, i.e., consider her trustworthy (*mechanism design by creditability*). The original game $G = (N, X, U)$ is modified to $G(V) := (N, X, [U + V])$ by these payments, where $[U + V]_i(x) = U_i(x) + V_i(x)$, that is, each player i obtains the payments of V_i in addition to the payoffs of U_i. The players' choice of strategies changes accordingly: Each player now selects a non-dominated strategy in $G(V)$. Henceforth, the set of non-dominated strategy profiles of $G(V)$ is denoted by $X^*(V)$, and $V(x)$ denotes the sum of all payments offered to the players when x is the game's outcome, $V(x) = \sum_{i=1}^n V_i(x)$. Observe that we have made two implicit assumptions: The mechanism designer can observe the actions chosen by the players and the players can determine the payoffs of all their strategies and compute the best strategy among them.

The mechanism designer's objective is to bring the players to choose a certain strategy profile, or a set of strategy profiles without spending too much. It is often cheaper for a mechanism designer to allow for entire region implementations rather than focusing on a fixed singleton profile. We consider two scenarios leading to two kinds of implementation cost: *worst-case implementation cost* and *uniform implementation cost*.

We first study a perfect knowledge scenario where all players know all strategy spaces X and payoff functions U, and the mechanism designer calculates with the maximum possible payments for a desired outcome (*worst-case implementation cost*). For a desired strategy profile set O, we say that payments V *implement* O if $\emptyset \subset X^*(V) \subseteq O$. V is called (worst-case) *k-implementation* if, in addition $V(x) \leq k$, $\forall\, x \in X^*(V)$. That is, the players' non-dominated strategies are within the desired strategy profile, and the payments do not exceed k for any possible outcome. Moreover, V is an *exact k-implementation* of O if all strategies of O are non-dominated in the resulting game, i.e., $X^*(V) = O$ and $V(x) \leq k\ \forall\, x \in X^*(V)$. The cost $k(O)$ of implementing O is the lowest of all non-negative numbers q for which there exists a q-implementation. If an implementation meets this lower bound, it is optimal, i.e., V is an *optimal implementation* of O if V implements O and $\max_{x \in X^*(V)} V(x) = k(O)$. The cost $k^*(O)$ of implementing O exactly is the smallest non-negative number q for which there exists an exact q-implementation of O. V is an *optimal exact implementation* of O if it implements O exactly and requires cost $k^*(O)$. The set of all implementations of O will be denoted by $\mathcal{V}(O)$, and the set of all exact implementations of O by $\mathcal{V}^*(O)$. Finally, a strategy profile set $O = \{z\}$ of cardinality one — consisting of only one strategy profile — is called a *singleton*. Clearly, for singletons it holds that non-exact and exact k-implementations are equivalent. For simplicity's sake we often write z instead of $\{z\}$. Observe that only subsets of X which are in $2^{X_1} \times 2^{X_2} \times \cdots \times 2^{X_n}$, i.e., the Cartesian product of subsets of the players' strategies, can be implemented exactly. We call such a subset of X a *rectangular strategy profile set*.[1] In conclusion, for the worst-case implementation cost, we have the following definitions.

Definition 1 (Worst-Case Cost and Exact Worst-Case Cost). The *worst-case implementation* cost of a strategy profile set O is denoted by $k(O) := \min_{V \in \mathcal{V}(O)}\{\max_{z \in X^*(V)} V(z)\}$. A strategy profile set O has *exact worst-case implementation* cost $k^*(O) := \min_{V \in \mathcal{V}^*(O)}\{\max_{z \in X^*(V)} V(z)\}$.

In a second scenario, we assume that a player i is aware of all strategy spaces X, but the player only knows her own utilities U_i rather than all players' utilities U. Without having any indication of what the others will play we presume a player chooses one of the non-dominated strategies uniformly at random. As a consequence, all strategy profiles in the non-dominated region $X^*(V)$ have the same probability of being picked and the mechanism designer can calculate an expected implementation cost. (An equivalent model would be a setting where the mechanism designer is less anxious, and makes the simplifying assumption that players sample the strategy rather than going for the worst-case.) We define the uniform

[1]Note that within our model where payments are made to individual players in different profiles, non-dominated profile sets will always be of rectangular shape.

cost of an implementation V as the *average* of all strategy profiles' possible cost in $X^*(V)$.

Definition 2 (Uniform Cost and Exact Uniform Cost). A strategy profile set O has *uniform implementation cost* $k_{UNI}(O) := \min_{V\in\mathcal{V}(O)}\{\text{avg}_{z\in X^*(V)} V(z)\}$ where avg is defined as $\text{avg}_{x\in X} f(x) := 1/|X| \cdot \sum_{x\in X} f(x)$. A strategy profile set O has *exact uniform implementation cost* $k^*_{UNI}(O) := \min_{V\in\mathcal{V}^*(O)}\{\text{avg}_{z\in X^*(V)} V(z)\}$.

1.2.3. *Leverage*

With rational players, mechanism designers can implement any desired outcomes if they offer high enough payments. The natural question that arises from this insight is for which games it actually makes sense to take influence at all, and which behavior the mechanism designer should implement in order to maximize her own utility. To answer this question we need to model the mechanism designer herself, and define the interests she has in the outcome of the game. In this work, we examine two diametrically opposed models of interested third parties. The first one is *benevolent* towards the participants of the game, and the other one *malicious*. While the former is interested in increasing a game's social gain, the latter seeks to minimize the players' welfare.[2] We define a measure indicating whether the mechanism of implementation enables them to modify a game in a favorable way such that their gain exceeds the manipulation's cost. We call these measures the *leverage* and *malicious leverage*, respectively. In the following, we will often write "(malicious) leverage" signifying both leverage and malicious leverage.

As the concept of leverage depends on the implementation cost, we examine the *worst-case* and the *uniform* leverage. The worst-case leverage is a lower bound on the mechanism designer's influence: We assume that without the additional payments, the players choose a strategy profile in the original game where the social gain is maximal, while in the modified game, they select a strategy profile among the newly non-dominated profiles where the difference between the social gain and the mechanism designer's cost is minimized. The value of the leverage is given by the net social gain achieved by this implementation minus the amount of money the mechanism designer had to spend. For malicious mechanism designers we have to invert signs and swap max and min. Moreover, the payments made by the mechanism designer have to be subtracted twice, because for a malicious mechanism designer, the money received by the players are considered a loss.

Definition 3 (Worst-Case Leverage). The *leverage* of a strategy profile set O is $LEV(O) := \max\{0, lev(O)\}$, where

$$lev(O) := \max_{V\in\mathcal{V}(O)}\left\{\min_{z\in X^*(V)}\{U(z) - V(z)\}\right\} - \max_{x^*\in X^*} U(x^*).$$

[2]Note that our terminology assumes the perspective of the players, i.e., if a mechanism designer acts contrary to their utilities, it is called "malicious". Depending on the game, a malicious mechanism designer's goal to punish the players can be morally upright (as illustrated in the introductory example).

Here $U(z)$ refers to the total utility of the players in profile z and $V(z)$ is the total amount of payments.

Definition 4 (Malicious Worst-Case Leverage). The *malicious leverage* of a strategy profile set O is $MLEV(O) := \max\{0, mlev(O)\}$, where

$$mlev(O) := \min_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)} \left\{ \max_{z \in X^*(V)} \{U(z) + 2V(z)\} \right\}.$$

Observe that according to our definitions, leverage values are always non-negative, as a mechanism designer has no incentive to manipulate a game if she will lose money. If the desired set consists only of one strategy profile z, i.e., $O = \{z\}$, we will speak of the *singleton* leverage. Similarly to the (worst-case) leverage, we define the uniform leverage.

Definition 5 (Uniform Leverage). The *uniform leverage* of a strategy profile set O is defined as $LEV_{UNI}(O) := \max\{0, lev_{UNI}(O)\}$, where

$$lev_{UNI}(O) := \max_{V \in \mathcal{V}(O)} \left\{ \operatorname*{avg}_{z \in X^*(V)} (U(z) - V(z)) \right\} - \operatorname*{avg}_{x^* \in X^*} U(x^*).$$

Definition 6 (Malicious Uniform Leverage). The *malicious uniform leverage* of a strategy profile set O is $MLEV_{UNI}(O) := \max\{0, mlev_{UNI}(O)\}$, where

$$mlev_{UNI}(O) := \operatorname*{avg}_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)} \left\{ \operatorname*{avg}_{z \in X^*(V)} \{U(z) + 2V(z)\} \right\}.$$

We define the *exact* (*uniform*) *leverage* $LEV^*(O)$ and the *exact* (*uniform*) *malicious leverage* $MLEV^*(O)$ by simply changing $\mathcal{V}(O)$ to $\mathcal{V}^*(O)$ in the definition of $LEV_{(UNI)}(O)$ and in the definition of $MLEV_{(UNI)}(O)$. Thus, the exact (uniform) (malicious) leverage measures a set's leverage if the interested party may only promise payments which implement O exactly. Finally, the (uniform) (malicious) leverages of an entire game $G = (N, X, U)$ are defined as the (uniform) (malicious) leverages of X, e.g., $LEV(G) := LEV(X)$.

1.3. *Organization*

This article is organized in two major sections. Section 2 investigates implementation *costs* and its computation complexity and we present algorithms for finding incentive compatible implementations of a desired set of outcomes. Section 3 then discusses the concept of leverage in games. We analyze the leverage complexities and present algorithms for computing the "potential" of such game manipulations. The article concludes with a discussion of the contributions.

2. Implementation Cost

The notion of k-implementations is introduced in Monderer and Tennenholtz [2003] to denote mechanisms that manipulate the players' behavior with payments of total value at most k. For the smallest implementable units of a game, singletons, they derived a closed formula for the minimal costs k needed to implement it. This formula builds on the fact that in order to implement a strategy profile $z \in X$, for each player i, strategy z_i must be the dominant strategy for i in the game $G(V)$ that combines the original payoffs with the offered payments. To achieve dominance $U_i(z) + V_i(z)$ must be at least as large as any payoff $U_i(x_i, z_{-i})$ of any other strategy $x_i \in X_i$, all other payments $V_i(z_i, x_{-i})$ can be chosen high enough to yield $U_i(z_i, x_{-i}) + V_i(z_i, x_{-i}) > U_i(x_i, x_{-i})$ for all $x_i \neq z_i, x_{-i} \neq z_{-i}$.

Theorem 1 (Monderer and Tennenholtz [2003]). *Let $G = (N, X, U)$ be a game with at least two strategies for every player. Every strategy profile z has an implementation V, and its implementation cost amounts to*

$$k(z) = \sum_{i=1}^{n} \max_{x_i \in X_i} (U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})).$$

Furthermore, observe that z constitutes a Nash equilibrium if and only if it holds for every player $i \in N$, $\max_{x_i \in X_i}(U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})) = 0$. As a corollary to Theorem 1 we get that a strategy profile z is a Nash equilibrium if and only if z has a 0-implementation. This remarkable result by Monderer and Tennenholtz [2003] implies that some outcomes can be implemented without spending anything. For a discussion of exact 0-implementations of profile sets, we refer the reader to Eidenbenz *et al.* [2007b].

Note that in general there are strategy profile regions for which it is cheaper to implement the entire region rather than a singleton within that region. Hence, it is worthwhile for a mechanism designer not to be too restrictive in what should be implemented. For example, if several outcomes are acceptable (and not just a singleton), better implementations may exist (e.g., in the game depicted in Fig. 2).

$G =$

20, 0	11, 9	15, 15	15, 15
11, 9	20, 0	15, 15	15, 15
19, 10	10, 19	9, 11	0, 20
10, 19	19, 10	0, 20	9, 11

$V =$

0, ∞	0, ∞	0, 0	0, 0
0, ∞	0, ∞	0, 0	0, 0
1, 1	1, 1	∞, 0	∞, 0
1, 1	1, 1	∞, 0	∞, 0

Fig. 2. 2-player game where O's optimal implementation V yields a region $|X^*(V)| > 1$. Each singleton o in the region O consisting of the four bottom left profiles has cost $k(o) = 11$ whereas V implements O at cost 2. This example can be generalized to an arbitrarily large difference in the implementation cost between a singleton and a region in the worst case.

2.1. *Worst case implementation cost*

We begin by studying exact implementations where the mechanism designer aims at implementing an *entire* strategy profile region. Exact region implementations are computationally cheaper to find compared to general region implementations, as calculating and comparing all the possible subregions is time-consuming. Subsequently, we examine general k-implementations.

2.1.1. *Exact implementation*

Recall that the matrix V is an exact k-implementation of a strategy region O iff $X^*(V) = O$ and $\sum_{i=1}^{n} V_i(x) \leq k \; \forall\, x \in X^*(V)$, i.e. each strategy O_i is part of the set of player i's non-dominated strategies for all Players i. We present the first correct algorithm to find such implementations.

Algorithm and Complexity. Recall that in our model each player classifies the strategies available to her as either dominated or non-dominated. Thereby, each dominated strategy $x_i \in X_i \backslash X_i^*$ is dominated by at least one non-dominated strategy $x_i^* \in X_i^*$. In other words, a game determines for each player i a relation M_i^G from dominated to non-dominated strategies, $M_i^G : X_i \backslash X_i^* \to X_i^*$, where $M_i^G(x_i) = x_i^*$ states that $x_i \in X_i \backslash X_i^*$ is dominated by $x_i^* \in X_i^*$. See Fig. 3 for an example.

When implementing a strategy profile region O exactly, the mechanism designer creates a modified game $G(V)$ with a new relation $M_i^V : X_i \backslash O_i \to O_i$ such that all strategies outside O_i map to at least one strategy in O_i. Therewith, the set of all newly non-dominated strategies of player i must constitute O_i. As every $V \in \mathcal{V}^*(O)$ determines a set of relations $M^V := \{M_i^V : i \in N\}$, there must be a set M^V for every V implementing O optimally as well. If we are given such an optimal relation set M^V without the corresponding optimal exact implementation, we can compute a V with minimal payments and the same relation M^V, i.e., given an optimal relation we can find an optimal exact implementation. As an illustrating example, assume an optimal relation set for G with $M_i^G(x_{i1}^*) = o_i$ and $M_i^G(x_{i2}^*) = o_i$. Thus, we can compute V such that o_i must dominate x_{i1}^* and x_{i2}^* in $G(V)$, namely, the condition $U_i(o_i, o_{-i}) + V_i(o_i, o_{-i}) \geq \max_{s \in (x_{i1}^*, x_{i2}^*)}(U_i(s, o_{-i}) + V_i(s, o_{-i}))$ must hold $\forall\, o_{-i} \in O_{-i}$. In an optimal implementation, Player i is not offered payments for strategy profiles of the form

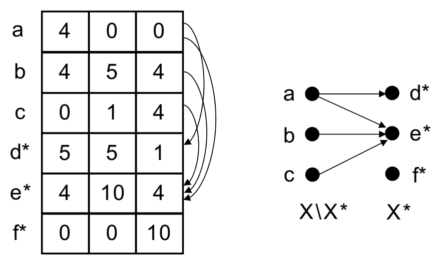

Fig. 3. Game from a single player's point of view with the corresponding relation of dominated ($X_i \backslash X_i^* = \{a, b, c\}$) to non-dominated strategies ($X_i^* = \{d^*, e^*, f^*\}$).

$(\bar{o}_i, x_{-i})$ where $\bar{o}_i \in X_i \setminus O_i$, $x_{-i} \in X_{-i}$. Hence, the condition above can be simplified to $V_i(o_i, o_{-i}) = \max(0, \max_{s \in \{x^*_{i1}, x^*_{i2}\}}(U_i(s, o_{-i}))) - U_i(o_i, o_{-i})$. Let $S_i(o_i) := \{s \in X_i \setminus O_i | M_i^V(s) = o_i\}$ be the set of strategies where M^V corresponds to an optimal exact implementation of O. Then, an implementation V with $V_i(\bar{o}_i, x_{-i}) = 0$, $V_i(o_i, \bar{o}_{-i}) = \infty$ for any Player i, and $V_i(o_i, o_{-i}) = \max\{0, \max_{s \in S_i(o_i)}(U_i(s, o_{-i}))\} - U_i(o_i, o_{-i})$ is an optimal exact implementation of O as well. Therefore, the problem of finding an optimal exact implementation V of O corresponds to the problem of finding an optimal set of relations $M_i^V : X_i \setminus O_i \rightarrow O_i$.

Our algorithm $\mathcal{ALG}_{\text{exact}}$ (cf. Algorithm 1) exploits this fact and constructs an implementation V for all possible relation sets, checks the cost that V would entail, and returns the lowest cost found. The computation is done for one player after the other, recursively. Note that V has reference semantics in Algorithm 1.

Algorithm 1 Exact k-Implementation ($\mathcal{ALG}_{\text{exact}}$)

Input: Game G, rectangular region O with $O_{-i} \subset X_{-i} \forall\ i$
Output: $k^*(O)$

1: $V_i(x) := 0$, $W_i(x) := 0\ \forall x \in X\ ,\ i \in N$;
2: $V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N,\ o_i \in O_i\ ,\ \bar{o}_{-i} \in X_{-i} \setminus O_{-i}$;
3: compute X^*;
4: **return** ExactK(V, n);

ExactK(V, i):
Input: payments V, current player i
Output: $k^*(O)$ for $G(V)$

1: **if** $|X_i^*(V) \setminus O_i| > 0$ **then**
2:     $s :=$ any strategy in $X_i^*(V) \setminus O_i$; $k_{best} := \infty$;
3:     **for all** $o_i \in O_i$ **do**
4:         **for all** $o_{-i} \in O_{-i}$ **do**
5:             $W_i(o_i, o_{-i}) := \max(0, U_i(s, o_{-i}) - (U_i(o_i, o_{-i}) + V_i(o_i, o_{-i})))$;
6:         $k :=$ ExactK($V + W$, i);
7:         **if** $k < k_{best}$ **then**
8:             $k_{best} := k$;
9:         **for all** $o_{-i} \in O_{-i}$ **do**
10:             $W_i(o_i, o_{-i}) := 0$;
11:     **return** k_{best};
12: **else if** $i > 1$ **then**
13:     **return** $ExactK(V, i - 1)$;
14: **else**
15:     **return** $\max_{o \in O} \sum_i V_i(o)$;

Theorem 2. $\mathcal{ALG}_{\text{exact}}$ *computes a strategy profile region's optimal exact implementation cost in time*

$$\mathrm{O}\left(|X|^2 \max_{i \in N}(|O_i|^{n|X_i^* \setminus O_i| - 1}) + n|O| \max_{i \in N}(|O_i|^{n|X_i^* \setminus O_i|})\right).$$

Note that $\mathcal{ALG}_{\text{exact}}$ has a large time complexity. In fact, a faster algorithm for this problem, called *Optimal Perturbation Algorithm* has been presented in Monderer and Tennenholtz [2003]. In a nutshell, this algorithm proceeds as follows: After initializing V similarly to our algorithm, the values of the region O in the matrix V are increased slowly for every Player i, i.e., by all possible differences between a player's payoffs in the original game. The algorithm terminates as soon as all strategies in $X_i^* \backslash O_i$ are dominated. Unfortunately, this algorithm does not always return an optimal implementation. Sometimes, it increases the values unnecessarily. An example demonstrating that the optimal perturbation algorithm presented in Monderer and Tennenholtz [2003] is not correct is the following game G with X^* and O and payments V_{OPT}, $V_{PERTURB}$.

G

2 / 0	0 / 0
0 / 0	2 / 3
4 / 0	0 / 0

V_{OPT}

2 / 0	5 / 0
0 / 3	5 / 0
0 / 5	0 / 0

$V_{PERTURB}$

2 / 3	5 / 0
2 / 3	5 / 0
0 / 5	0 / 0

As can be verified easily, V_{OPT} implements O with cost $k = 3$. The matrix $V_{PERTURB}$ computed by the optimal perturbation algorithm implements O as well, however, it has cost $k = 5$.

Not only does this leave us without a polynomial algorithm, we even conjecture that the problem is inherently hard and that deciding whether an k-exact implementation exists is **NP**-hard. Although we did not succeed in proving **NP**-hardness we have reason to believe so as we can show the arguably easier, and closely related problem of finding the exact uniform implementation cost of a strategy region to be **NP**-hard (Theorem 3).

Conjecture 1. *Finding an optimal exact implementation of a strategy region is* **NP***-hard.*

The study of exact implementation cost was introduced by Monderer and Tennenholtz [2003] primarily because it seems easier to compute the exact implementation cost of a region O than its non-exact cost. Computing O's non-exact cost implicitly computes at least the optimal subregion's exact cost, potentially the exact cost of all subsets of O since the algorithm has to discover that no other subregion has lower implementation cost. Unfortunately, although we experienced that computing exact cost is computationally easier than computing non-exact cost, it still seems infeasible to do so in polynomial time.

2.1.2. *Non-exact implementation*

In contrast to exact implementations where the complete set of strategy profiles O must be non-dominated, the additional payments in non-exact implementations only have to ensure that a *subset* of O is the newly non-dominated region. Obviously, it matters which subset this is. Knowing that a subset $O' \subseteq O$ bears optimal cost, we could find $k(O)$ by computing $k^*(O')$. As we conjectured that computing exact cost is in **NP** we get the following:

Conjecture 2. *Finding an optimal implementation of a strategy region is* **NP**-*hard.*

Apart from the fact that finding an optimal implementation includes solving the — believed to be **NP**-hard — optimal exact implementation cost problem for at least one subregion of O, finding this subregion might also be **NP**-hard even if the exact implementation cost problem shows to be in **P** since there are exponentially many possible subregions. In fact, a reduction from the SAT problem is presented in Monderer and Tennenholtz [2003]. The authors show how to construct a 2-person game in polynomial time given a CNF formula such that the game has a 2-implementation if and only if the formula has a satisfying assignment. However, their proof is not correct: While there indeed exists a 2-implementation for every satisfiable formula, it can be shown that 2-implementations also exist for non-satisfiable formulas. E.g., strategy profiles $(x_i, x_i) \in O$ are always 1-implementable. Unfortunately, we were not able to correct their proof. However, we conjecture the problem to be **NP**-hard, i.e., we assume that no algorithm can do much better than performing a brute force computation of the exact implementation cost (cf. Algorithm 1) of all possible subsets, unless **NP** = **P**. Note that we give a reduction from SET COVER for the uniform implementation cost in the following section.

2.2. *Uniform implementation cost*

In the uniform model, we assume non-dominated strategy profiles are played with the same probability. This assumption is reasonable in settings where players have imperfect knowledge and only know their own utility function rather than all players' utilities. Without any indication of what the others will play, it seems a player's natural strategy to mix among the non-dominated pure strategies uniformly at random yielding a uniform probability distribution over the non-dominated strategy profiles. Note that this assumption can be modeled either on the level of the players or on the level of the mechanism designer. We either presume the players to adopt a certain behavior or we presume the mechanism designer to make some assumptions on the players' behavior. The argument supporting the uniform assumption stated above reasons on the level of the players' behavior. To reason on the latter level we could think of the mechanism designer as willing to take risks and presume her to anticipate uniform rather than worst case costs regardless of the scope of information available to the players.

In the following we show that it is **NP**-hard to compute the uniform implementation cost for both the non-exact and the exact case. We devise game configurations

	e_1	e_2	e_3	e_4	e_5	d	r
e_1	5	0	0	0	0	1	0
e_2	0	5	0	0	0	1	0
e_3	0	0	5	0	0	1	0
e_4	0	0	0	5	0	1	0
e_5	0	0	0	0	5	1	0
s_1	5	0	0	5	0	0	0
s_2	0	5	0	5	0	0	0
s_3	0	5	5	0	5	0	0
s_4	5	5	5	0	0	0	0

Fig. 4. Payoff matrix for player 1 in a game which reduces the SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U} = \{e_1, e_2, e_3, e_4, e_5\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, $S_1 = \{e_1, e_4\}$, $S_2 = \{e_2, e_4\}$, $S_3 = \{e_2, e_3, e_5\}$, $S_4 = \{e_1, e_2, e_3\}$ to the problem of computing $k^*_{UNI}(O)$. The optimal exact implementation V of O in this sample game adds a payment V_1 of 1 to the strategy profiles (s_1, d) and (s_3, d), implying that the two sets S_1 and S_3 cover $\mathcal{U}$ optimally.

which reduce SET COVER to the problem of finding an implementation of a strategy profile set with optimal uniform cost.

Theorem 3. *In games with at least two players, the problem of finding a strategy profile set's* exact uniform implementation cost *is* **NP**-*hard.*

Proof. For a given universe $\mathcal{U}$ of l elements $\{e_1, e_2, \ldots, e_l\}$ and m subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, with $S_i \subset \mathcal{U}$, SET COVER is the problem of finding the minimal collection of S_i's which contains each element $e_i \in \mathcal{U}$. We assume without loss of generality that $\nexists (i \neq j) : S_i \subset S_j$. Given a SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$, we can efficiently construct a game $G = (N, X, U)$ where $N = \{1, 2\}$, $X_1 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$, and $X_2 = \{e_1, e_2, \ldots, e_l, d, r\}$. Each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. Player 1's payoff function U_1 is defined as follows: $U_1(e_i, e_j) := m + 1$ if $i = j$ and 0 otherwise, $U_1(s_i, e_j) := m + 1$ if $e_j \in S_i$ and 0 otherwise, $U_1(e_i, d) := 1$, $U_1(s_i, d) := 0$, $U_1(x_1, r) := 0 \; \forall x_1 \in X_1$. Player 2 has a payoff of 0 when playing r and 1 otherwise. In this game, strategies e_j are not dominated for Player 1 because in column d, $U_1(e_j, d) > U_1(s_i, d)$, $\forall i \in \{1, \ldots, m\}$. The set O we would like to implement is $\{(x_1, x_2) | x_1 = s_i \wedge (x_2 = e_i \vee x_2 = d)\}$. See Fig. 3 for an example. Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$, where each Q_j corresponds to an S_i. We now claim that Q is an optimal solution for a SET COVER problem, an optimal exact implementation V of O in the corresponding game has payments $V_1(s_i, d) := 1$ if $Q_i \in Q$ and 0 otherwise, and all payments $V_1(s_i, e_j)$ equal 0.

Note that by setting $V_1(s_i, d)$ to 1, strategy s_i dominates all strategies e_i which correspond to an element in S_i. Thus, our payment matrix makes all strategies e_i

of Player 1 dominated since any strategy e_i representing element e_i is dominated by the strategies s_j corresponding to S_j which cover e_i in the minimal covering set. (If $|S_j| = 1$, s_j gives only equal payoffs in $G(V)$ to those of e_i in the range of O_2. However, s_j can be made dominating e_i by increasing s_j's payments $V_1(s_j, r)$ in the extra column r.) If there are any strategies s_i dominated by other strategies s_j, we can make them non-dominated by adjusting the payments $V_1(s_i, r)$ for column r. Hence, any solution of SC corresponds to a valid exact implementation of O.

It remains to show that such an implementation is indeed optimal and there are no other optimal implementations not corresponding to a minimal covering set. Note that by setting $V_1(s_i, d) := 1$ and $V_1(s_i, r) > 0$ for all s_i, all strategies e_j are guaranteed to be dominated and V implements O exactly with uniform cost $\text{avg}_{o\in O} V(o) = m/|O|$. If an implementation had a positive payment for any strategy profile of the form (s_i, e_j), it would cost at least $m + 1$ to have an effect. However, a positive payment greater than m yields a larger. Thus, an optimal V has positive payments inside set O only in column d. By setting $V_1(s_i, d)$ to 1, s_i dominates the strategies e_j which correspond to the elements in S_i, due to our construction. An optimal implementation has a minimal number of 1s in column d. This can be achieved by selecting those rows s_i ($V_1(s_i, d) := 1$), which form a minimal covering set and as such all strategies e_i of Player 1 are dominated at minimal cost. Our reduction can be generalized for $n > 2$ by simply adding players with only one strategy and zero payoffs in all strategy profiles. □

Theorem 4. *In games with at least three players, the problem of finding a strategy profile set's* non-exact uniform implementation cost *is* **NP**-*hard.*

Proof. We give a similar reduction of SET COVER to the problem of computing $k_{UNI}(O)$ by extending the setup we used for proving the exact case. We add a third player and show **NP**-hardness for $n = 3$ first and indicate how the reduction can be adapted for games with $n > 3$. Given a SET COVER problem instance $SC = (\mathcal{U}, \mathcal{S})$, we can construct a game $G = (N, X, U)$ where $N = \{1, 2, 3\}$, the strategies for the players are $X_1 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$, $X_2 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m, d, r\}$ and $X_3 = \{a, b\}$. Again, each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. In the following, we use '_' in profile vectors as a placeholder for any possible strategy. Player 1's payoff function U_1 is defined as follows: $U_1(e_i, e_j, \_) := (m + l)^2$ if $i = j$ and 0 otherwise, $U_1(e_i, s_j, \_) := 0$, $U_1(s_i, e_j, \_) := (m+l)^2$ if $e_j \in S_i$ and 0 otherwise, $U_1(s_i, s_j, \_) := 0$ if $i = j$ and $(m + l)^2$ otherwise, $U_1(e_i, d, \_) := 1$, $U_1(s_i, d, \_) := 0$, $U_1(\_, r, \_) := 0$. Player 2 has a payoff of $(m + l)^2$ for any strategy profile of the form $(s_i, s_i, \_)$ and 0 for any other strategy profile. Player 3 has a payoff of $m + l + 2$ for strategy profiles of the form (s_i, s_i, b), a payoff of 2 for profiles (s_i, e_i, b) and profiles $(s_i, s_j, b), i \neq j$, and a payoff of 0 for any other profile. The set O we would like to implement is $\{(x_1, x_2, x_3)|x_1 = s_i \wedge (x_2 = e_i \vee x_2 = s_i \vee x_2 = d) \wedge (x_3 = a)\}$. See Fig. 5 for an example. First, note the fact that any implementation of O will

	e_1	e_2	e_3	e_4	e_5	s_1	s_2	s_3	s_4	d	r
e_1	81, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	1, 0	0, 0
e_2	0, 0	81, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	1, 0	0, 0
e_3	0, 0	0, 0	81, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	1, 0	0, 0
e_4	0, 0	0, 0	0, 0	81, 0	0, 0	0, 0	0, 0	0, 0	0, 0	1, 0	0, 0
e_5	0, 0	0, 0	0, 0	0, 0	81, 0	0, 0	0, 0	0, 0	0, 0	1, 0	0, 0
s_1	81, 0	0, 0	0, 0	81, 0	0, 0	0, 81	81, 0	81, 0	81, 0	0, 0	0, 0
s_2	0, 0	81, 0	0, 0	81, 0	0, 0	81, 0	0, 81	81, 0	81, 0	0, 0	0, 0
s_3	0, 0	81, 0	81, 0	0, 0	81, 0	81, 0	81, 0	0, 81	81, 0	0, 0	0, 0
s_4	81, 0	81, 0	81, 0	0, 0	0, 0	81, 0	81, 0	81, 0	0, 81	0, 0	0, 0

	e_1	e_2	e_3	e_4	e_5	s_1	s_2	s_3	s_4	d	r
e_1	0	0	0	0	0	0	0	0	0	0	0
e_2	0	0	0	0	0	0	0	0	0	0	0
e_3	0	0	0	0	0	0	0	0	0	0	0
e_4	0	0	0	0	0	0	0	0	0	0	0
e_5	0	0	0	0	0	0	0	0	0	0	0
s_1	2	2	2	2	2	11	2	2	2	0	0
s_2	2	2	2	2	2	2	11	2	2	0	0
s_3	2	2	2	2	2	2	2	11	2	0	0
s_4	2	2	2	2	2	2	2	2	11	0	0

Fig. 5. Payoff matrix for Players 1 and 2 given Player 3 chooses a and payoff matrix for Player 3 when she plays strategy b in a game which reduces a SET COVER instance $SC = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U} = \{e_1, e_2, e_3, e_4, e_5\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, $S_1 = \{e_1, e_4\}, S_2 = \{e_2, e_4\}, S_3 = \{e_2, e_3, e_5\}, S_4 = \{e_1, e_2, e_3\}$ to the problem of computing $k_{UNI}(O)$. Every implementation V of O in this game needs to add any positive payment in the second matrix to V_3, i.e. $V_3(x_1, x_2, a) = U_3(x_1, x_2, b)$, in order to convince Player 3 of playing strategy a. An optimal implementation adds a payment V_1 of 1 to the strategy profiles (s_1, d, a) and (s_3, d, a), implying that the two sets S_1 and S_3 cover $\mathcal{U}$ optimally in the corresponding SET COVER problem.

have $V_3(o_1, o_2, a) \geq U_3(o_1, o_2, b)$, in order to leave Player 3 no advantage playing b instead of a. In fact, setting $V_3(o_1, o_2, a) = U_3(o_1, o_2, b)$ suffices. (Setting any $V_3(a, \bar{o}_{-3}) > U_3(b, \bar{o}_{-3})$ where $\bar{o}_{-3}$ is outside O lets Player 3 choose strategy a.) Also note that for Player 2, O_2 can be made non-dominated without offering any payments inside O, e.g., set $V_2(e_i, e_j, \_) = 1$ and $V_2(e_i, d, \_) = 1$.

Analogously to the exact case's proof, we claim that iff $Q = \{Q_1, Q_2, \ldots, Q_k\}$, where each Q_j corresponds to an S_i, is an optimal solution for a SET COVER problem, there exists an optimal exact implementation V of O in the corresponding game. This implementation selects a row s_i ($V_1(s_i, d, a) = 1$), if $Q_i \in Q$ and does not select s_i ($V_1(s_i, d, a) = 0$) otherwise. All other payments V_1 inside O are 0. Player 2's payments $V_2(o)$ are 0 for all $o \in O$ and player 3's payoffs are set to $V_3(o_1, o_2, a) = U_3(o_1, o_2, b)$. A selected row s_i contributes $\text{cost}_{s_i} = (3(l+m)+1)/(l+m+1)$. A non-selected row s_j contributes $\text{cost}_{s_j} = (3(l+m))/(l+m+1) < \text{cost}_{s_i}$. Thus including non-selected rows in $X^*(V)$ can be profitable. Selecting all rows s_i yields a correct implementation of O with uniform cost $\text{avg}_{i=1}^{m} \text{cost}_{s_i} = (3(l+m)+1)/(l+m+1) < 3$.

In fact, the game's payoffs are chosen such that it is not worth implementing any set smaller than O. We show for every set smaller than O, that its exact uniform implementation cost is strictly larger. Assume a set yielding lower cost implements α strategies for Player 1, β strategies e_i and γ strategies s_j for Player 2. Note that implementing Player 2's strategy d is profitable if $\beta + \gamma > 0$, as it adds α to the denominator and at most α to the numerator of the implementation cost of sets without d. Consequently, there are three cases to consider: (i) $\beta \neq 0, \gamma = 0$: The costs add up to $\sum_{o \in O}(V_1(o) + V_2(o) + V_3(o))/|O| \geq (1 + (m+l)^2 + 2\alpha\beta)/(\alpha(\beta+1))$, which is greater than 3, since $\alpha \leq m, \beta \leq l$. (ii) $\beta = 0, \gamma \neq 0$: The aggregated cost is at least $(1 + \alpha(m+l) + 2\alpha\gamma)/(\alpha(\gamma+1))$, which is also greater than 3.

(iii) $\beta \neq 0, \gamma \neq 0$: Assume there are κ sets necessary to cover U. Hence the sum of the payments in column d is at least κ. In this case, the cost amounts to $(\kappa + \alpha(m+l) + 2\alpha(\beta+\gamma))/(\alpha(\beta+\gamma+1)) = 2 + (m+l-2+\kappa/\alpha)/(\beta+\gamma+1) \geq k^*(O)$. Equality only holds if $\alpha = \gamma = m$ and $\beta = l$. We can conclude that O has to be implemented exactly in order to obtain minimal cost.

Therefore, an optimal implementation yields $X^*(V) = O$ with the inalienable payments to Player 3 and a minimal number of 1-payments to Player 1 for strategy profiles (s_i, d, a) such that every e_j is dominated by at least one s_i. The number of 1-payments is minimal if the selected rows correspond to a minimal covering set, and the claim follows.

Note that a similar SET COVER reduction can be found for games with more than three players. Simply add players to the described 3-player game with only one strategy. □

Due to the nature of the reduction the inapproximability results of SET COVER (Alon *et al.* [2006]; Feige [1998]) carry over to our problem.

Theorem 5. *Unless* $\mathbf{P} = \mathbf{NP}$ *the best approximation ratio a polynomial-time algorithm can achieve is* $\Omega(n \max_i\{\log|X_i^* \backslash O_i|\})$ *for both the exact and non-exact implementation cost within any function of the input length.*

Proof. *Exact Case*: In order to prove this claim, a reduction similar to the one in the proof of Theorem 3 can be applied. Consider again a SET COVER instance with a universe $\mathcal{U}$ of l elements $\{e_1, e_2, \ldots, e_l\}$ and m subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, with $S_j \subset \mathcal{U}$. We construct a game $G = (N, X, U)$ with n players $N = \{1, \ldots, n\}$, where $X_i = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}\ \forall\, i \in \{1, \ldots, n-1\}$, and $X_n = \{e_1, e_2, \ldots, e_l, d, r\}$. Again, each strategy e_j corresponds to an element $e_j \in \mathcal{U}$, and each strategy s_j corresponds to a set S_j. Player i's payoff function U_i, for $i \in \{1, \ldots, n-1\}$, is defined as follows: Let e_k and s_k be strategies of Player i and let e_l be a strategy of Player n. If $k = l$, player i has payoff $m+1$, and 0 otherwise. Moreover, $U_i(s_k, e_l) := m+1$ if $e_l \in S_k$ and 0 otherwise, and $U_i(e_k, d) := 1$, $U_i(s_k, d) := 0$, $U_i(x_k, r) := 0\ \forall\, x_k \in X_i$. Thus, player i's payoffs only depend on Player i and Player n's strategies. Player n has a payoff of 0 when playing r and 1 otherwise, independently of all other players' choices. We ask for an implementation of set O where Player i, for $i \in \{1, \ldots, n-1\}$, plays any strategy s_k, and Player n plays any strategy e_l or strategy d.

Due to the independence of the players' payoffs, the situation is similar to the example in Fig. 3, and a SET COVER instance has to be solved for each Player i $\forall\, i \in \{1, \ldots, n-1\}$. According to the well-known inapproximability results for SET COVER, no polynomial time algorithm exists which achieves a better approximation ratio than $\Omega(\log|X_i^* \backslash O_i|)$ for each Player i, unless $\mathbf{P} = \mathbf{NP}$, and the claim follows.

Non-Exact Case: We use the inapproximability results for SET COVER again. Concretely, we assume a set of $n = 3k$ players for an arbitrary constant $k \in \mathbb{N}$

and make k groups of three players each. The payoffs of the three players in each group are the same as described in the proof of Theorem 4 for the non-exact case, independently of all other players' payoffs. Hence, SET COVER has to be solved for $n/3$ players. □

Remark 1. The uniform implementation cost is based on the assumption that players choose one of the non-dominated strategies uniformly at random such that an equal probability mass is assigned to each strategy profile in the non-dominated region. I.e., the implementation cost depends on the aggregate cost over the entire profile set. This enables us to construct a game corresponding to a set cover problem instance. In the worst-case model however, individual strategy profiles need to be taken into account and payment differences between strategy profiles matter. Put differently, the worst-case model assumes less about the players behavior than the uniform model. We believe that this renders the minimal implementation cost problem only harder. Therefore, we conjecture that the worst case implementation cost is **NP**-hard as well.

3. Leverage

This section studies to which extent the social welfare of a game can be influenced by a mechanism designer within economic reason, i.e., by taking the implementation cost into account. To this end we study the *leverage*, a measure indicating whether the mechanism of implementation allows to modify a game in a favorable way such that the gain exceeds the manipulation's cost, as defined in Sec. 1.2. Besides considering classic, benevolent mechanism designers, we also analyze malicious mechanism designers seeking to minimize the players' welfare. For instance, we show that a malicious mechanism designer can sometimes corrupt games and worsen the players' situation to a larger extent than the amount of money invested.

3.1. *Worst-case leverage*

We first study singleton implementations and then extend our investigations to profile sets.

3.1.1. *Singletons*

In the following we will examine a mechanism designer seeking to implement a game's best singleton, i.e., the strategy profile with the highest singleton leverage. Dually, a malicious designer attempts to find the profile of the largest malicious leverage.

We propose an algorithm that computes two arrays, LEV and $MLEV$, containing all (malicious) singletons' leverage within a strategy profile set O. By setting $O = X$, the algorithm computes all singletons' (malicious) leverage of a game. We make use of a formula presented in Monderer and Tennenholtz [2003] for computing

Algorithm 2 Singleton (Malicious) Leverage

Input: Game G, Set $O \subseteq X$

Output: LEV and $MLEV$

compute X^*;
for all strategy profiles $x \in O$ **do**
  $lev[x] := -\max_{x^* \in X^*} U(x^*)$;
  $mlev[x] := \min_{x^* \in X^*} U(x^*)$;
for all Players $i \in N$ **do**
  for all $x_{-i} \in O_{-i}$ **do**
    $m := \max_{x_i \in X_i} U_i(x_i, x_{-i})$;
    for all strategies $z_i \in O_i$ **do**
      $lev[z_i, x_{-i}]$ += $2 \cdot U_i(z_i, x_{-i}) - m$;
      $mlev[z_i, x_{-i}]$ += $U_i(z_i, x_{-i}) - 2m$;
$\forall\, o \in O$: $LEV[o] := \max\{0, lev[o]\}$;
$\forall\, o \in O$: $MLEV[o] := \max\{0, mlev[o]\}$;
return LEV, $MLEV$;

a singleton's cost, namely that $k(z) = \sum_{i=1}^{n} \max_{x_i \in X_i}\{U_i(x_i, z_{-i}) - U_i(z_i, z_{-i})\}$. Algorithm 2 initializes the lev-array with the negative value of the original game's maximal social gain in the non-dominated set and the $mlev$-array with its minimal social gain. Next, it computes the set of non-dominated strategy profiles X^*; in order to do this, we check, for each player and for each of her strategies, whether the strategy is dominated by some other strategy. In the remainder, the algorithm adds up the players' contributions to the profiles' (malicious) leverage for each player and strategy profile. In any field z of the leverage array lev, we add the amount that Player i would contribute to the social gain if z was played and subtract the cost we had to pay her, namely $U_i(x_i, x_{-i}) - (m - U_i(x_i, x_{-i})) = 2U_i(x_i, x_{-i}) - m$. For any entry z in the malicious leverage array $mlev$, we subtract player i's contribution to the social gain and also twice the amount the designer would have to pay if z is played since she loses money and the players gain it, $-U_i(x_i, x_{-i}) - 2(m - U_i(x_i, x_{-i})) = U_i(x_i, x_{-i}) - 2m$. Finally, lev and $mlev$ will contain all singletons' leverage and singletons' malicious leverage in O. By replacing the negative entries by zeros, the corresponding leverage arrays LEV and $MLEV$ are computed. The interested party can then lookup the best *non-negative* singleton by searching the maximal entry in the respective array.

Theorem 6. *For a game where every player has at least two strategies, Algorithm* 2 *computes all singletons*' (*malicious*) *leverage within a strategy profile set O in* $\mathrm{O}(n|X|^2)$ *time.*

Proof. The correctness of Algorithm 2 follows directly from the application of the (malicious) singleton leverage formula. It remains to prove the time complexity. Finding the non-dominated strategies in the original game requires time

$\sum_{i=1}^{n} \binom{|X_i|}{2} |X_{-i}| = O(n|X|^2)$, and finding the maximal or minimal *gain* amongst the possible outcomes X^* of the original game requires time $O(n|X|)$. The time for all other computations can be neglected asymptotically, and the claim follows. □

3.1.2. *Strategy profile sets*

Observe that implementing singletons may be optimal for entire strategy sets as well, namely in games where the strategy profile set yielding the largest (malicious) leverage is of cardinality 1. In some games, however, dominating all other strategy profiles in the set is expensive and unnecessary. Therefore, a mechanism designer is bound to consider sets consisting of more than one strategy profile as well to find a subset of X yielding the maximal (malicious) leverage. Moreover, we can construct games where the difference between the best (malicious) set leverage and the best (malicious) singleton leverage gets arbitrarily large. Figure 6 depicts such a game.

$G =$

α, 0	1, 0	γ, 0	γ, 0
1, 0	α, 0	γ, 0	γ, 0
$\alpha - 1$, 0	0, $\alpha - 1$	0, α	0, 1
0, $\alpha - 1$	$\alpha - 1$, 0	0, 1	0, α

$V_O =$

0, ∞	0, ∞	0, 0	0, 0
0, ∞	0, ∞	0, 0	0, 0
1, 1	1, 1	∞, 0	∞, 0
1, 1	1, 1	∞, 0	∞, 0

$V_s =$

0, ∞	0, 0	0, 0	0, 0
0, ∞	0, 0	0, 0	0, 0
0, α	0, 0	0, 0	0, 0
α, 1	0, α	∞, 0	∞, 0

Fig. 6. Two-player game where the set O bears the largest leverage. Implementation V_O yields $X^*(V_O) = O$ and V_s yields one non-dominated strategy profile. By offering payments V_O, a mechanism designer has cost 2, no matter which $o \in O$ will be played. However, she changes the social welfare to $\alpha - 1$. If $\gamma < \alpha - 3$ then O has a leverage of $\alpha - 3 - \gamma$ and if $\gamma > \alpha + 3$ then O has a malicious leverage of $\gamma - \alpha - 3$. Any singleton $o \in O$ has an implementation cost of $\alpha + 1$, yet the resulting leverage is 0 and the malicious leverage is $\max(0, \gamma - 3\alpha - 1)$. This demonstrates that a profile set O's (malicious) leverage can be arbitrarily large, even if its singletons's (malicious) leverage is zero.

Although many factors influence a strategy profile set's (malicious) leverage, there are some simple observations. First, if rational players already choose strategies such that the strategy profile with the highest social gain is non-dominated, a designer will not be able to ameliorate the outcome. Just as well, a malicious interested party will have nothing to corrupt if a game already yields the lowest social gain possible.

Fact 7. (i) If a game G's social optimum $x_{opt} := \arg\max_{x \in X} U(x)$ is in X^* then $LEV(G) = 0$. (ii) If a game G's social minimum $x_{worst} := \arg\min_{x \in X} U(x)$ is in X^* then $MLEV(G) = 0$.

As an example, a class of games where both properties (i) and (ii) of Fact 7 always hold are *equal sum games*, where every strategy profile yields the same gain, $U(x) = c \; \forall x \in X, c : constant$. (Zero sum games are a special case of equal sum games where $c = 0$.)

Fact 8 (Equal Sum Games). The leverage and the malicious leverage of an equal sum game G is zero: $LEV(G) = 0$, $MLEV(G) = 0$.

A well-known example of an zero sum game is *Matching Pennies*: Two players each secretly turn a penny to heads or tails. Then they reveal their choices simultaneously. If both coins show the same face, Player 2 gives his penny to Player 1; if the pennies do not match, Player 2 gets the pennies. This matching pennies game features another interesting property: There is no dominated strategy. Therefore an interested party could only implement strategy profile sets O which are subsets of X^*. This raises the question whether a set $O \subseteq X^*$ can ever have a (malicious) leverage. We find that the answer is no and moreover:

Theorem 9. *The leverage of a strategy profile set $O \subseteq X$ intersecting with the set of non-dominated strategy profiles X^* is $(M)LEV = 0$.*

Proof. Assume that $|O \cap X^*| > 0$ and let $\hat{z}$ be a strategy profile in the intersection of O and X^*. Let $x^*_{\max} := \arg\max_{x^* \in X^*} U(x^*)$ and $x^*_{\min} := \arg\min_{x^* \in X^*} U(x^*)$. Let V_{LEV} be any implementation of O reaching $LEV(O)$ and V_{MLEV} any implementation of O reaching $MLEV(O)$. We get for the leverage $LEV(O) = \max\{0, \min_{z \in X^*(V_{LEV})}\{U(z) - V_{LEV}(z)\} - U(x^*_{\max})\} \leq \max\{0, [U(\hat{z}) - V_{LEV}(\hat{z})] - U(x^*_{\max})\} \leq \max\{0, U(x^*_{\max}) - V_{LEV}(\hat{z}) - U(x^*_{\max})\} = \max\{0, -V_{LEV}(\hat{z})\} = 0$, and for the malicious leverage $MLEV(O) = \max\{0, U(x^*_{\min}) - \max_{z \in X^*(V_{MLEV})}[U(z) + 2V_{MLEV}(z)]\} \leq \max\{0, U(x^*_{\min}) - U(\hat{z}) - 2V_{MLEV}(\hat{z})\} \leq \max\{0, U(x^*_{\min}) - U(x^*_{\min}) - 2V_{MLEV}(\hat{z})\} = \max\{0, -2V_{MLEV}(\hat{z})\} = 0$. □

In general, the problem of computing a strategy profile set's (malicious) leverage seems computationally hard. It is related to the problem of computing a set's implementation cost, which we conjectured in Sec. 2 to be **NP**-hard, and hence, we conjecture the problem of finding $LEV(O)$ or $MLEV(O)$ to be **NP**-hard in general as well. In fact, we can show that computing the (malicious) leverage has at least the same complexity as computing a set's cost.

Theorem 10. *If the computation of a set's implementation cost is* **NP**-*hard the computation of a strategy profile set's* (*malicious*) *leverage is also* **NP**-*hard.*

Proof. We proceed by reducing the problem of computing $k(O)$ to the problem of computing $MLEV(O)$. Theorem 9 allows us to assume that O and X^* do not intersect since $O \cap X^* \neq \emptyset$ implies $MLEV(O) = 0$. By definition, a strategy profile set's cost are $\min_{V \in \mathcal{V}(O)}\{\max_{z \in X^*(V)} V(z)\}$ and from the malicious leverage's definition, we have $\min_{V \in (V)}\{\max_{z \in X^*(V)}\{U(z)+2V(z)\}\} = \min_{x^* \in X^*} U(x^*) - mlev(O)$. The latter equation's left-hand side almost matches the formula for $k(O)$ if not for the term $U(z)$ and a factor of 2. If we can modify the given game such that all strategy profiles inside $X^*(V) \subseteq O$ have a gain of $\gamma := -2n \max_{x \in X}\{\max_{i \in N} U_i(x)\} - \min_{x^* \in X^*} U(x^*) - \epsilon$ where $\epsilon > 0$, we will be able to reduce O's cost to $k(O) = (\min_{x^* \in X^*} U(x^*) - mlev(O) - \gamma)/2 = (-mlev(O) + 2n \max_{x \in X}\{\max_{i \in N} U_i(x)\} + \epsilon)$, thus $mlev(O) > 0$ and $MLEV(O) = mlev(O)$, ensuring that $MLEV(O)$ and $mlev(O)$ are polynomially reducible to each other. This is achieved by the following transformation of a problem instance (G, O) into a problem instance (G', O): Add an additional Player $n+1$ with one strategy a and a payoff function $U_{n+1}(x)$ equal to $\gamma - U(x)$ if $x \in O$ and 0 otherwise. Thus, a strategy profile x in G' has social gain equal to γ if it is in O and equal to $U(x)$ in the original game if it is outside O. As Player $n+1$ has only one strategy available, G' has the same number of strategy profiles as G and furthermore, there will be no payments V_{n+1} needed in order to implement O. Player $(n+1)$'s payoffs impact only the profiles' gain, and they have no effect on how the other players decide their tactics. Thus, the non-dominated set in G' is the same as in G and it does not intersect with O. Since the transformation does not affect the term $\min_{x^* \in X^*} U(x^*)$, the set's cost in G are equal to $(\min_{x^* \in X^*} U(x^*) - MLEV(O) - \gamma)/2$ in G'.

Reducing the problem of computing $k(O)$ to $lev(O)$ is achieved by using the same game transformation where an additional player is introduced such that $\forall\, o \in O : U(o) = \gamma$, where $\gamma := n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} + \max_{x^* \in X^*}\{U(x^*)\} + \epsilon$ for $\epsilon > 0$. We can then simplify the leverage formula to $lev(O) = \gamma - k(O) - \max_{x^* \in X^*} U(x^*) = n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} - k(O) + \epsilon > 0$ and thus we find the cost $k(O)$ by computing $n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} - LEV(O) - \epsilon$. □

The task of finding a strategy profile set's leverage is computationally hard. Recall that we have to find an implementation V of O which maximizes the term $\min_{z \in X^*(V)}\{U(z) - V(z)\}$. Thus, there is at least one implementation $V \in \mathcal{V}(O)$ bearing O's leverage. Since this V implements a subset of O exactly, it is also valid to compute O's leverage by searching among all subsets O' of O the one with the largest exact leverage $LEV^*(O')$. (Note that we do not provide algorithms for computing the malicious leverage but for the benevolent leverage only. However, we

1st Reading

July 19, 2011 16:17 WSPC/0219-1989 151-IGTR 00282

Algorithm 3 Exact Leverage

Input: Game G, rectangular set O with $O_{-i} \subset X_{-i} \forall\ i$
Output: $LEV^*(O)$

$V_i(x) := 0$, $W_i(x) := 0\ \forall x \in X\ ,\ i \in N$;
$V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N$, $o_i \in O_i\ ,\ \bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
compute X_i^*;
return $\max\{0, ExactLev(V, n) - \max_{x^* \in X^*} U(x^*)\}$;

ExactLev$(V,\ i)$**:**

Input: payments V, current player i
Output: $lev^*(O)$ for $G(V)$

if $|X_i^*(V) \backslash O_i| > 0$ **then**
  $s :=$ any strategy in $X_i^*(V) \backslash O_i$; $lev_{best} := 0$;
  for all $o_i \in O_i$ **do**
    for all $o_{-i} \in O_{-i}$ **do**
      $W_i(o_i, o_{-i}) := \max\{0, U_i(s, o_{-i}) - (U_i(o_i, o_{-i}) + V_i(o_i, o_{-i}))\}$;
    $lev := ExactLev(V + W, i)$;
    if $lev > lev_{best}$ **then**
      $lev_{best} := lev$;
    for all $o_{-i} \in O_{-i}$ **do**
      $W_i(o_i, o_{-i}) := 0$;
  return lev_{best};
if $i > 1$ **return** $ExactLev(V, i - 1)$;
else return $\min_{o \in O}\{U(o) - V(o)\}$;

are sure that a malicious mechanism designer will figure out how to adapt our algorithms for the benevolent leverage for a nastier purpose.)

In the following we will provide an algorithm which computes a rectangular strategy profile set's exact leverage. It makes use of the fact that if $X^*(V)$ has to be a subset of O, each strategy $\bar{o}_i \notin O_i$ must be dominated by at least one strategy o_i in the resulting game $G(V)$ — a property which has been observed and exploited before in the previous section in order to compute a set's exact cost. In order to compute $LEV(O)$, we can apply Algorithm 3 for all rectangular subsets and return the largest value found.

Theorem 11. *Algorithm* 3 *computes a strategy profile set's exact leverage in time*

$$\mathrm{O}\left(|X|^2 \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|-1}) + n|O| \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|})\right).$$

Proof. Clearly, the algorithm is correct as it searches for all possibilities of a strategy in $X_i \backslash O_i$ to be dominated by a strategy in O_i. The time complexity follows from solving the doubly recursive equation over the strategy set and the number of players (compare to the analysis of Algorithm 1 in the previous section). □

3.2. *Uniform leverage*

In the setting where a mechanism designer applies uniform implementations the players have less information of the game and are assumed to play a non-dominated strategy uniformly at random. This allows her to calculate with the average cost and thus, the observation stating that the uniform (malicious) leverage is always at least as large as the worst-case (malicious) leverage does not surprise.

Theorem 12. *A set's uniform (malicious) leverage is always larger than or equal to the set's (malicious) leverage.*

Proof.

$$\begin{aligned} lev_{UNI}(O) &= \max_{V\in\mathcal{V}(O)}\left\{\operatorname*{avg}_{z\in X^*(V)}\{U(z)-V(z)\}\right\} - \operatorname*{avg}_{x^*\in X^*(V)} U(x^*) \\ &\geq \max_{V\in\mathcal{V}(O)}\left\{\min_{z\in X^*(V)}\{U(z)-V(z)\}\right\} - \max_{x^*\in X^*(V)} U(x^*) \\ &= lev(O) \end{aligned}$$

and

$$\begin{aligned} mlev_{UNI}(O) &= \operatorname*{avg}_{x^*\in X^*(V)} U(x^*) - \min_{V\in\mathcal{V}(O)}\left\{\operatorname*{avg}_{z\in X^*(V)}\{U(z)+2V(z)\}\right\} \\ &\geq \min_{x^*\in X^*(V)}\{U(x^*)\} - \min_{V\in\mathcal{V}(O)}\left\{\max_{z\in X^*(V)}\{U(z)+2V(z)\}\right\} \\ &= mlev(O). \end{aligned}$$

□

Another difference concerns the sets O intersecting with X^*, i.e., $O \cap X^* \neq \emptyset$: Unlike the worst-case leverage (Theorem 9), the uniform leverage can exceed zero in these cases, as can be verified by calculating O's leverage in Fig. 3.

3.2.1. *Complexity*

We show how the uniform implementation cost can be computed in polynomial time given the corresponding leverage. Thus the complexity of computing the leverage follows from the **NP**-hardness of finding the optimal implementation cost. The lower bounds are derived by modifying the games constructed from the SET COVER problem in Theorem 3, and by using a lower bound for the approximation quality of the SET COVER problem. If no polynomial time algorithm can approximate the size of a set cover within a certain factor, we get an arbitrarily small approximated leverage $LEV_{UNI}^{approx} \leq \epsilon$ while the actual leverage is large. Hence the approximation ratio converges to infinity and, unless $\mathbf{P} = \mathbf{NP}$, there exists no polynomial time algorithm approximating the leverage of a game within any function of the input length.

Theorem 13. *For games with at least two players, the problem of*

- *computing a strategy profile set's* exact uniform leverage *as well as*
- *computing its* exact malicious uniform leverage *are* **NP**-*hard.*

For games with at least three players, the problem of

- *computing a strategy profile set's* non-exact uniform leverage *as well as*
- *computing its* non-exact malicious uniform leverage *are* **NP**-*hard.*

Furthermore, the (exact) uniform leverage of O cannot be approximated in polynomial time within any function of the input length unless $\mathbf{P} = \mathbf{NP}$.

Proof. **NP**-*Hardness*: *Exact Case.* The claim follows from the observation that if $(M)LEV^*_{UNI}(O)$ is found, we can immediately compute $k^*_{UNI}(O)$ which is **NP**-hard (Theorem 3). Due to the fact that any $z \in O$ is also in $X^*(V)$ for any $V \in \mathcal{V}^*(O)$ we know that

$$\begin{aligned} lev^*_{UNI}(O) &= \max_{V \in \mathcal{V}^*(O)} \left\{ \operatorname*{avg}_{z \in X^*(V)} \{U(z) - V(z)\} \right\} - \operatorname*{avg}_{z \in X^*} U(x^*) \\ &= \max_{V \in \mathcal{V}^*(O)} \left\{ \operatorname*{avg}_{z \in O} U(z) - \operatorname*{avg}_{z \in O} V(z) \right\} - \operatorname*{avg}_{x^* \in X^*} U(x^*) \\ &= \operatorname*{avg}_{z \in O} U(z) - \min_{V \in \mathcal{V}^*(O)} \left\{ \operatorname*{avg}_{z \in O} V(z) \right\} - \operatorname*{avg}_{x^* \in X^*} U(x^*) \\ &= \operatorname*{avg}_{z \in O} U(z) - \mathbf{k^*_{UNI}(O)} - \operatorname*{avg}_{x^* \in X^*} U(x^*), \text{ and} \\ mlev^*_{UNI}(O) &= \operatorname*{avg}_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}^*(O)} \left\{ \operatorname*{avg}_{z \in X^*(V)} \{U(z) + 2V(z)\} \right\} \\ &= \operatorname*{avg}_{x^* \in X^*} U(x^*) - \operatorname*{avg}_{z \in O} U(z) - 2 \min_{V \in \mathcal{V}^*(O)} \left\{ \operatorname*{avg}_{z \in O} V(z) \right\} \\ &= \operatorname*{avg}_{x^* \in X^*} U(x^*) - \operatorname*{avg}_{z \in O} U(z) - 2\mathbf{k^*_{UNI}(O)}. \end{aligned}$$

Observe that $\operatorname{avg}_{x^* \in X^*} U(x^*)$ and $\operatorname{avg}_{z \in O} U(z)$ can be computed easily. Moreover, as illustrated in the proof of Theorem 10, we can efficiently construct a problem instance (G', O) from any (G, O) with the same cost, such that for G': $(m)lev_{(UNI)} = (M)LEV_{(UNI)}$.

Non-Exact Case. The claim can be proved by reducing the **NP**-hard problem of computing $k_{UNI}(O)$ to the problem of computing $(M)LEV_{UNI}(O)$. This reduction uses a slight modification of player 3's utility in the respective game in the proof of Theorem 3 ensuring $\forall z \in O \; U(z) = \gamma := -4(m + l)^2 - 2m^2 + m(l + m)$.

Set $U_3(s_i, e_j, a) = \gamma - U_1(s_i, e_j, a) - U_2(s_i, e_j, a)$, $U_3(s_i, e_j, b) = \gamma + 2 - U_1(s_i, e_j, a) - U_2(s_i, e_j, a)$ for all $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, l\}$, $U_3(s_i, s_j, a) = \gamma - U_1(s_i, s_j, a) - U_2(s_i, s_j, a)$, $U_3(s_i, s_j, b) = \gamma + 2 - U_1(s_i, s_j, a) - U_2(s_i, s_j, a)$ for all $i \neq j$, $U_3(s_i, s_i, a) = \gamma - U_1(s_i, s_i, a) - U_2(s_i, s_i, a)$, $U_3(s_i, s_i, b) = \gamma + (m + l + 2) - U_1(s_i, s_i, a) - U_2(s_i, s_i, a)$ for all i. Since in this 3-player game, $mlev_{UNI}(O) > 0$, we can give a formula for $k_{UNI}(O)$ depending only on O's (malicious) leverage and the average social gain, namely $k_{UNI}(O) = (\text{avg}_{x^* \in X^*} U(x^*) - MLEV_{UNI}(O))/2$. Thus, once $MLEV_{UNI}(O)$ is known, $k_{UNI}(O)$ can be computed immediately, and therefore finding the uniform malicious leverage is **NP**-hard as well. We can adapt this procedure for $LEV_{UNI}(O)$ as well.

Lower Bound Approximation: *Exact Case.* The game constructed from the SET COVER problem in Theorem 3 for the exact case is modified as follows: The utilities of Player 1 remain the same. The utilities of Player 2 are all zero except for $U_2(e_1, r) = (l + m)(\sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - k\mathcal{LB} - \epsilon)$, where k is the minimal number of sets needed to solve the corresponding SET COVER instance, $\epsilon > 0$, and $\mathcal{LB}$ denotes a lower bound for the approximation quality of the SET COVER problem. Observe that X^* consists of all strategy profiles of column r. The target set we want to implement exactly is given by $O_1 = \{s_1, \ldots, s_m\}$ and $O_2 = \{e_1, \ldots, e_l, d\}$. We compute $lev_{UNI}^{\text{opt}} = \text{avg}_{o \in O} U(o) - \text{avg}_{x \in X^*} U(x) - k = \sum_{i=1}^{m} |S_i|(m+1)/(ml + m) - \sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - (-k\mathcal{LB} - \epsilon) - k = k(\mathcal{LB} - 1) + \epsilon$. As no polynomial time algorithm can approximate k within a factor $\mathcal{LB}$, $LEV_{UNI}^{\text{approx}} \leq \epsilon$. Since $\lim_{\epsilon \to 0} LEV_{UNI}^{\text{opt}} / LEV_{UNI}^{\text{approx}} = \infty$ the claim follows for a benevolent mechanism designer.

For malicious mechanism designers, we modify the utilities of the game from the proof of Theorem 5 for Player 2 as follows: $U_2(e_1, r) = (l + m)(2k\mathcal{LB} + \epsilon + \sum_{i=1}^{m} |S_i|(m+1)/(ml+m))$, and $U_2(\_, \_) = 0$ for all other profiles. It is easy to see that by a similar analysis as performed above, the theorem also follows in this case.

Non-Exact Case. We modify the game construction of Theorem 3's proof for the non-exact case by setting $U_2(e_1, r, b) := ((\sum_{i=1}^{m} |S_i|(m + l)^2 + m^2(m + l)^2 + 3m(m + l))/(m^2 + ml + m) - k\mathcal{LB} - \epsilon)(m + l)$, where k is the minimal number of sets needed to solve the corresponding SET COVER instance, $\epsilon > 0$, and $\mathcal{LB}$ denotes a lower bound for the approximation quality of the SET COVER problem and zero otherwise. Observe that $X^* = \{x | x \in X, x = (\_, r, b)\}$, O has not changed, and payments outside O do not contribute to the implementation cost; therefore, implementing O exactly is still the cheapest solution. By a similar analysis as in the proof of Theorem 3 the desired result is attained. For malicious mechanism designers we set $U_2(e_1, r, b) = ((\sum_{i=1}^{m} |S_i|(m+l)^2 + m^2(m+l)^2 + 3m(m+l))/(m^2 + ml + m) + 2k\mathcal{LB} + \epsilon)(m + l)$ and proceed as above. □

3.2.2. *Algorithms*

To find algorithms that compute the uniform leverage we can adapt the algorithms for the worst-case leverage from Sec. 3.1. Recall Algorithm 2 that computes the leverage of singletons of a desired strategy profile set. We can adapt Lines 3 and 4 to accommodate the definition of the uniform leverage, i.e., set $mlev[x] := \text{avg}_{x^* \in X^*} U(x^*)$ and $mlev[x] := -mlev[x]$. The resulting algorithm helps finding an optimal singleton.

A benevolent mechanism designer can adapt Algorithm 3 in order to compute $LEV^*_{UNI}(O)$: She only has to change Line 4 to $\max\{0, ExactLev(V, n) - \text{avg}_{x^* \in X^*} U(x^*)\}$ and 'min' in Line 13 to 'avg'. Invoking this algorithm for any $O' \subseteq O$ yields the subset O with maximal leverage $LEV_{UNI}(O)$.

Due to Theorem 13 there is no polynomial time algorithm giving a non-trivial approximation of a uniform leverage. The simplest method to find a lower bound for $LEV_{UNI}(O)$ is to search the singleton in O with the largest uniform leverage. Unfortunately, there are games (cf. Fig. 2) where this lower bound is arbitrarily bad, analogously to lower bound for the worst case leverage.

4. Conclusions and Outlook

This article has addressed the problem of how to influence the behavior of players (e.g., to improve cooperation) in contexts where it is, e.g., not possible to specify interaction rules, for example, in computer networks. We studied a mechanism designer that manipulates outcomes by creditability, i.e., by promising payments, and studied the natural questions: Which outcomes can be implemented by promising players money? What is the corresponding cost? By introducing the concept of leverage, we analyzed which outcomes are worth implementing and computed the corresponding gains formally. We have presented algorithms for various objectives yielding implementations of low cost, as well as computational complexity results for worst-case games and games with imperfect knowledge and mixed strategies. We have also initiated the study of benevolent and malicious mechanism designers intending to change the game's outcome if the improvement or deterioration in social welfare exceeds the implementation cost. Our results are summarized in Figs. 7 and 8.

Implementation Cost	Complexity	Properties
Uniform	NP-complete SINGLETON $O(n \cdot \sum_i \lvert X_i \rvert)$ ZERO $O(n\lvert X\rvert^2)$	NE 0-implementable
Worst-case	conjecture: NP-complete SINGLETON $O(n \cdot \sum_i \lvert X_i \rvert)$ ZERO $O(n\lvert X\rvert^2)$	NE 0-implementable

Fig. 7. Complexity results for the computation of the implementation cost. Unless stated otherwise, complexities refer to the problem of computing any strategy profile's implementation cost. SINGLETON indicates the complexity of computing a singleton's implementation cost. ZERO indicates the complexity of deciding for a strategy profile region whether it is 0-implementable. The complexities of ZERO are results from our earlier work (see the COCOA conference version).

Leverage	Complexity	Properties
Uniform	NP-complete SINGLETON $O(n \cdot \sum_i \|X_i\|)$	$MLEV_{UNI} \geq MLEV$
Worst-case	as hard as implementation cost SINGLETON $O(n \cdot \sum_i \|X_i\|)$	$O \cap X^* \neq \emptyset \Rightarrow (M)LEV = 0$ social opt/worst $\in X^*$ $\Rightarrow (M)LEV = 0$ Equal-sum games $\Rightarrow (M)LEV = 0$

Fig. 8. Complexity results for the computation of the leverage. SINGLETON indicates the complexity of computing a singleton's leverage.

There exist several interesting directions for future research, including the quest for implementation cost approximation algorithms or for game classes which allow a leverage approximation. Furthermore, the mixed leverage and the leverage of dynamic games with an interested third party offering payments in each round are still unexplored.

References

Aiyer, A. S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.-P. and Porth, C. [2005] BAR fault tolerance for cooperative services, in *Proceedings of the 20th ACM Symposium on Operating Systems Principles* (*SOSP*), New York, NY, USA (ACM Press), 45–58.

Alon, N., Moshkovitz, D. and Safra, M. [2006] Algorithmic construction of sets for k-restrictions, *ACM Transactions on Algorithms* (*TALG*), 153–177.

Aspnes, J., Chang, K. and Yampolskiy, A. [2005] Inoculation strategies for victims of viruses and the sum-of-squares partition problem, in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), 43–52.

Babaioff, M., Feldman, M. and Nisan, N. [2006a] Combinatorial agency, in *Proceedings of the 7th ACM Conference on Electronic Commerce* (*EC*), 18–28.

Babaioff, M., Feldman, M. and Nisan, N. [2006b] Mixed strategies in combinatorial agency, in *Proceedings of the Internet and Network Economics, Second International Workshop* (*WINE*), 353–364.

Babaioff, M., Kleinberg, R. and Papadimitriou, C. H. [2007] Congestion games with malicious players, in *Proceedings of the 8th ACM Conference on Electronic Commerce* (*EC*), New York, NY, USA (ACM Press), 103–112. .

Bradonjic, M., Ercal-Ozkaya, G., Meyerson, A. and Roytman, A. [2009] On the price of mediation, in *Proceeding of the ACM Conference on Electronic Commerce* (*EC*), 315–324.

Christodoulou, G. and Koutsoupias, E. [2005] The price of anarchy of finite congestion games, in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing* (*STOC*), 67–73.

Cole, R., Dodis, Y. and Roughgarden, T. [2003a] How much can taxes help selfish routing? in *Proceedings of the 4th ACM Conference on Electronic Commerce* (*EC*), New York, NY, USA (ACM Press), 98–107.

Cole, R., Dodis, Y. and Roughgarden, T. [2003b] Pricing network edges for heterogeneous selfish users, in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing* (*STOC*), New York, NY, USA (ACM Press), 521–530.

Dash, R. K., Jennings, N. R. and Parkes, D. C. [2003, November] Computational-mechanism design: A call to arms, *IEEE Intelligent Systems*, Special Issue on Agents and Markets, 40–47.

Dybvig, P. and Spatt, C. [1983] Adoption externalities as public goods, *J. of Public Economics* **20**, 231–247.

Eidenbenz, R., Oswald, Y. A., Schmid, S. and Wattenhofer, R. [2007a] Manipulation in games, in *Proc. 18th International Symposium on Algorithms and Computation* (*ISAAC*).

Eidenbenz, R., Oswald, Y. A., Schmid, S. and Wattenhofer, R. [2007b] Mechanism design by creditability, in *Proceedings of the 1st International Conference on Combinatorial Optimization and Applications* (*COCOA*)*, Springer LNCS 4616*.

Eidenbenz, R. and Schmid, S. [2009] Combinatorial agency with audits, in *Proceedings of the IEEE International Conference on Game Theory for Networks* (*GameNets*).

Feige, U. [1998] A threshold of $\log n$ for approximating set cover, *Journal of the ACM*, 634–652.

Feigenbaum, J. and Shenker, S. [2003] Distributed algorithmic mechanism design: Recent results and future directions, Distributed Computing Column, Bulletin of the EATCS, 101–121.

Maskin, E. [1999] Review of economic studies, *Nash Equilibrium and Welfare Optimality*, 23–38.

Maskin, E. and Sjöström, T. [2002] *Handbook of Social Choice Theory and Welfare* (*Implementation Theory*) North-Holland, Amsterdam.

Monderer, D. and Tennenholtz, M. [2003] k-implementation, in *Proceedings of the 4th ACM Conference on Electronic Commerce* (*EC*), New York, NY, USA (ACM Press), 19–28.

Monderer, D. and Tennenholtz, M. [2009] Strong mediated equilibrium, *Artif. Intell.* **173**(1).

Moscibroda, T. and Schmid, S. [2009] On mechanism design without payments for throughput maximization, in *Proceedings of the 18th IEEE Conference on Computer Communication* (*INFOCOM*).

Moscibroda, T., Schmid, S. and Wattenhofer, R. [2006] When selfish meets evil: Byzantine players in a virus inoculation game, in *Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing* (*PODC*), New York, NY, USA (ACM Press), 35–44.

Nisan, N., Roughgarden, T., Tardos, E. and Vazirani, V. [2007] *Algorithmic Game Theory*, Cambridge.

Osborne, M. J. and Rubinstein, A. [1994] *A Course in Game Theory*, MIT Press.

Porter, R., Ronen, A., Shoham, Y. and Tennenholtz, M. [2008] Fault tolerant mechanism design, *Artif. Intell.* **172**(15), 1783–1799.

Resnick, E., Bachrach, Y., Meir, R. and Rosenschein, J. S. [2009] The cost of stability in network flow games, in *Proceedings of the Mathematical Foundations of Computer Science* (*MFCS*), 636–650.

Rosenschein, J. S. and Zlotkin, G. [1994] Rules of Encounter, *MIT Press*.

Roughgarden, T. [2001] Stackelberg scheduling strategies, in *Proceedings of the ACM Symposium on Theory of Computing* (*STOC*), 104–113.

Segal, I. [1999] Contracting with externalities, *The Quarterly Journal of Economics* **2**, 337–388.

Spiegler, R. [2000] Extracting intercation-created surplus, *Games and Economic Behavior* **30**, 142–162.
Zhou, X., Gandhi, S., Suri, S. and Zheng, H. [2008] eBay in the Sky: Strategy-proof wireless spectrum auctions, in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking* (*MOBICOM*), New York, NY, USA (ACM Press), 2–13.