Manipulation in Games*

Raphael Eidenbenz, Yvonne Anne Oswald,
Stefan Schmid, and Roger Wattenhofer

Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland

"Private Vices by the dextrous Management of a skilful Politician may be turned into Publick Benefits"

Fable of the Bees (B. Mandeville)

Abstract. This paper studies to which extent the social welfare of a game can be influenced by an interested third party within economic reason, i.e., by taking the implementation cost into account. Besides considering classic, benevolent mechanism designers, we also analyze malicious mechanism designers. For instance, this paper shows that a malicious mechanism designer can often corrupt games and worsen the players' situation to a larger extent than the amount of money invested. Surprisingly, no money is needed at all in some cases. We provide algorithms for finding the so-called leverage in games and show that for optimistic mechanism designers, computing the leverage or approximations thereof is **NP**-hard.

1 Introduction

Consider the following extension of the well-known prisoners' dilemma where two bank robbers, both members of the *Al Capone clan*, 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). 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

* Research supported in part by the Swiss National Science Foundation (SNF).

both remain silent, they will receive money compensating for one year in jail,[1] and furthermore, 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 clan an accumulated two years in jail.

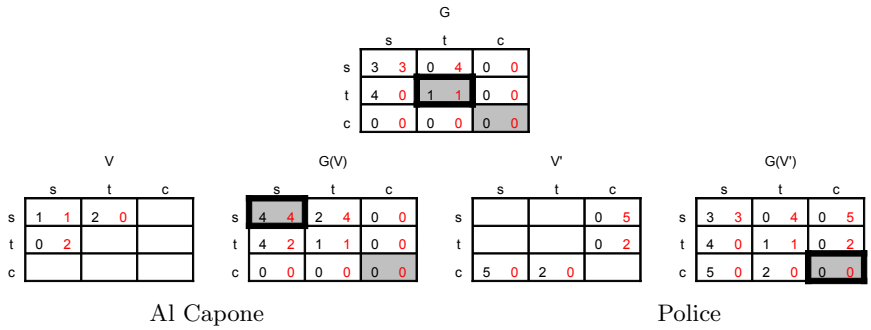

Fig. 1. Extended prisoners' dilemma: G shows the prisoners' initial payoffs, where payoff values equal saved years. The first strategy is to remain silent (s), the second to testify (t) and the third to confess (c). Nash equilibria are colored gray, and non-dominated strategy profiles have a bold border. 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.

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 clan. We say that the *leverage* of the strategy profile where both prisoners play s is two. For the police however, 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

[1] For this scenario, we presume that time really is money!

largest (malicious) leverage and therewith the problem of implementing the corresponding outcome at optimal cost. This paper analyzes these problems' complexities and presents algorithms for finding the leverage of games for cautious and optimistic mechanism designers. We show that while the leverage of a single strategy profile can be computed efficiently for both cautious and optimistic mechanism designers, finding an optimal implementation for a set of strategy profiles is **NP**-hard by a reduction from the SETCOVER problem, and we provide a lower bound for the approximation attainable by any polynomial-time algorithm. Moreover, we prove that an optimistic mechanism designer cannot compute the leverage of a game in polynomial time unless **P=NP** and finding approximations thereof is hard as well.

Related Work. Algorithmic game theory and mechanism design have become popular tools for gaining insights into the sociological, economical and political complexity of today's distributed systems such as politics, global markets or the Internet (we refer to [10,11] for an introduction). Typically, when a game-theoretic analysis reveals that a system may suffer from selfish behavior, appropriate countermeasures have to be taken in order to enforce a desired behavior (e.g. [7]).

As it is often infeasible for a mechanism designer to influence the rules according to which the players act in a distributed system, she has to resort to other measures. One way of manipulating the players' decision-making is to offer them money for certain outcomes. Monderer and Tennenholtz [8] showed how creditablility can be used to outwit selfish agents and influence their decisions; in some cases no money actually has to be paid at all to implement a certain behavior (cf. also [13]). The authors consider a mechanism designer who cannot enforce behaviors and cannot change the system, and who attempts to lead agents to adopt desired behaviors in a given multi-agent setting. The only way the third party can influence the outcome of the game is by promising non-negative monetary transfers conditioned on the observed behavior of the agents. Eidenbenz et al. [5] have continued the analysis of [8] and have provided deeper insights into the possibilities and algorithmic complexities of mechanism design based on creditability. They presented algorithms for computing a strategy profile set's implementation cost and extended the notion of k-implementation to round-based games, risk-averse player games and average payoff games. Moreover, they show that the complexity results given in [8] are not correct.

This paper extends [5,8] by introducing 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 costs. However, as a first contribution, in this paper, we show that a malicious mechanism designer can in some cases even *reduce* the social welfare at no costs. Second, we present algorithms to compute both the regular 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 paper 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* [1] has been

introduced which aims at understanding the impact of altruistic and malicious behavior in game theory. Moscibroda et al. [9] analyze a virus inoculation game consisting of both selfish and malicious players. A similar model has recently been studied in the context of congestion games [4].

Our work is also related to *Stackelberg theory* [12] which attends to games with selfish players where a fraction of the entire population is orchestrated by a global leader. In contrast to our paper, this leader is not bound to offer any incentives (or credits) to follow his objectives. In the recent research thread of *combinatorial agencies* [3], a setting is studied where a mechanism designer seeks to influence the outcome of a game by contracting the players individually; however, she cannot observe the player's individual actions and the contracts only depend on the overall outcome.

2 Model

First, we will review some game theory notation and the notion of k-implementation [8]. We will then introduce the concept of *leverage* and *malicious leverage* in games.

Game Theory. A *strategic game* can be described by a tuple $G = (N, X, U)$. $N = \{1, 2, \ldots, n\}$ 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 $X := X_1 \times X_2 \times \ldots \times X_n$. 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 $X_{-i} = X_1 \times \ldots \times X_{i-1} \times X_{i+1} \times \ldots \times X_n$. An element of X_i is denoted by x_i, and similarly, $x_{-i} \in X_{-i}$; hence x_{-i} is a vector consisting of the strategy profiles available if Player i selects Strategy x_i. $U = (U_1, U_2, \ldots, U_n)$ is an n-tuple of *payoff functions*, where $U_i : X \to \mathbb{R}$ determines Player i's payoff arising from the game's outcome. We will refer to the sum of the individual player's payoffs of a given strategy profile $x \in X$ as the strategy profile's *gain* $U(x) := \sum_{i=1}^{n} U_i(x)$.

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 $X^* = X_1^* \times \ldots \times X_n^*$ 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 $x \in X$ is a *Nash equilibrium* if no unilateral deviation in strategy by any single player is profitable, that is $\forall i \in N, U_i(x_i, x_{-i}) \geq U_i(x_i', x_{-i})$.

k-Implementation. We assume that players are rational and always choose a non-dominated strategy. Moreover, they do not cooperate. We examine the impact of payments to players offered by a *mechanism designer* (an interested third party) who seeks to influence the outcome of a game. These payments are described by a tuple of non-negative payoff functions $V = (V_1, V_2, \ldots, V_n)$, where $V_i : X \to \mathbb{R}^+$, i.e. the payments depend on the strategy Player i selects as well as on the choices of all other players. We assume that the players trust the mechanism designer to finally pay the promised amount of money,

i.e., consider her trustworthy. 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 payoff 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 $X^*(V)$. For a strategy profile x, the sum of the additional payments to all players is denoted by the *payment* $V(x) := \sum_{i=1}^{n} V_i(x)$. A *strategy profile set* $O \subseteq X$ of G is a subset of all strategy profiles X. Similarly to X_i and X_{-i}, we define $O_i := \{x_i | \exists x_{-i} \in X_{-i} \text{ s.t. } (x_i, x_{-i}) \in O\}$ and $O_{-i} := \{x_{-i} | \exists x_i \in X_i \text{ s.t. } (x_i, x_{-i}) \in O\}$. The mechanism designer's main objective is to force the players to choose a certain strategy profile or a set of strategy profiles, without spending too much. This paper studies two kinds of implementation costs: *worst-case implementation costs* and *uniform implementation costs*.

First, we will consider a pessimistic scenario where the mechanism designer calculates with the maximum possible payments for a desired outcome (*worst-case implementation costs*). 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 $X^*(V) = O$ 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 $2^{X_1} \times 2^{X_2} \times \ldots \times 2^{X_n}$, i.e., the Cartesian product of subsets of the players' strategies, can be implemented exactly. We call such a subset of X a *convex strategy profile set*.[2] In conclusion, for the worst-case implementation costs, we have the following definitions.

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

The assumption that the cost of an implementation V is equal to the cost of the strategy profile in $X^*(V)$ with the *highest* payments is pessimistic. This paper therefore also looks at a less anxious mechanism designer who takes the risk of high worst case costs if the expected costs are small. If players only

[2] These sets define a convex area in the n-dimensional hyper-cuboid, provided that the strategies are depicted such that all o_i are next to each other.

know their own utilities, assuming them to select one of their non-dominated strategies uniformly at random, is a first simple model an optimistic mechanism designer might apply. We define the uniform cost of an implementation V as the *average* of all strategy profiles' possible cost in $X^*(V)$. Thus we assume all non-dominated strategy profiles $x \in X^*(V)$ to have the same probability.

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

(Malicious) Leverage. Mechanism designers can implement desired outcomes in games at certain costs. This raises the question for which games it makes sense to take influence at all. This paper examines two diametrically opposed kinds of interested parties, the first one being *benevolent* towards the participants of the game, and the other being *malicious*. While the former is interested in increasing a game's social gain, the latter seeks to minimize the players' welfare. 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. Note that in the following, we will often write "(malicious) leverage" signifying both leverage and malicious leverage.

As the concept of leverage depends on the implementation costs, 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 (Malicious) Leverage). *Let* $lev(O) := \max_{V \in \mathcal{V}(O)} \{\min_{z \in X^*(V)} \{U(z) - V(z)\}\} - \max_{x^* \in X^*} U(x^*)$ *and* $mlev(O) := \min_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)} \{\max_{z \in X^*(V)} \{U(z) + 2V(z)\}\}$. *The* leverage *and* malicious leverage *of a strategy profile set O are* $LEV(O) := \max\{0, lev(O)\}$ *and* $MLEV(O) := \max\{0, mlev(O)\}$, *respectively.*

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 can define the uniform leverage for less anxious mechanism designers.

Definition 4 (Uniform (Malicious) Leverage). *Let* $lev_{UNI}(O) := \max_{V \in \mathcal{V}(O)}\{\varnothing_{z \in X^*(V)}(U(z) - V(z))\} - \varnothing_{x^* \in X^*} U(x^*)$ *and* $mlev_{UNI}(O) := \varnothing_{x^* \in X^*} U(x^*) - \min_{V \in \mathcal{V}(O)}\{\varnothing_{z \in X^*(V)}\{U(z) + 2V(z)\}\}$. *The* uniform leverage *and* malicious uniform leverage *of a strategy profile set* O *are* $LEV_{UNI}(O) := \max\{0, lev_{UNI}(O)\}$ *and* $MLEV_{UNI}(O) := \max\{0, mlev_{UNI}(O)\}$, *respectively.*

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 $MLEV_{(UNI)}(O)$. Thus, the exact (uniform) (malicious) leverage measures a set's leverage if the interested party may only promise payments which implement O exactly.

3 Worst-Case Leverage

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 [8] for computing 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 1 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.

Algorithm 1 Singleton (Malicious) Leverage

Input: Game G, Set $O \subseteq X$
Output: LEV and $MLEV$

```
1: compute X*;
2: for all strategy profiles x ∈ O do
3:    lev[x] := − max_{x*∈X*} U(x*);
4:    mlev[x] := min_{x*∈X*} U(x*);
5: for all Players i ∈ N do
6:    for all x_{−i} ∈ O_{−i} do
7:       m := max_{x_i∈X_i} U_i(x_i, x_{−i});
8:       for all strategies z_i ∈ O_i do
9:          lev[z_i, x_{−i}] += 2 · U_i(z_i, x_{−i}) − m;
10:         mlev[z_i, x_{−i}] += U_i(z_i, x_{−i}) − 2m;
11: ∀ o ∈ O: LEV[o] := max{0, lev[o]};
12: ∀ o ∈ O: MLEV[o] := max{0, mlev[o]};
13: return LEV, MLEV;
```

Theorem 1. *For a game where every player has at least two strategies, Algorithm 1 computes all singletons' (malicious) leverage within a strategy profile set O in $\mathrm{O}\big(n|X|^2\big)$ time.*

Proof. The correctness of Algorithm 1 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 $\mathrm{O}(n\,|X|)$. The time for all other computations can be neglected asymptotically, and the claim follows. □

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. Fig. 2 depicts such a game.

$G =$

20	0	11	9	10	10	10	10
11	9	20	0	10	10	10	10
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. Two-player game where set O bears the largest leverage. Implementation V yields $X^*(V) = O$. By offering payments V, a benevolent mechanism designer has cost 2, no matter which $o \in O$ will be played. However, she improves the social welfare by 9. Thus O has a leverage of 7 whereas any singleton $o \in O$ has a leverage of 0. By reducing Player 2's payoffs in the upper game half and Player 1's payoffs in the right game half, O's leverage gets arbitrarily large.

A similar game can be used to show an arbitrarily large difference for the *malicious* leverage: E.g., set the payoffs in the four upper right strategy profiles of the game G in Fig. 2 to 100 instead of 10. V still implements O but switching to O now decreases the social gain.

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 2 *(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 2 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 3 (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 toss a penny. 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 4. *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 is conjectured in [5] 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 5. *If the computation of a set's implementation cost is* **NP***-hard, then 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 4 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 $k(O) = \min_{V \in \mathcal{V}(O)}\{\max_{z \in X^*(V)} V(z)\}$ 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 $\forall o \in O : U(o) = \gamma := n \max_{x \in X}\{\max_{i \in N}\{U_i(x)\}\} + \max_{x^* \in X^*}\{U(x^*)\} + \epsilon$ 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 $LEV^*(O')$.[3]

Algorithm 2 Exact Leverage

Input: Game G, convex set O with $O_{-i} \subset X_{-i} \forall\ i$
Output: $LEV^*(O)$
1: $V_i(x) := 0,\ W_i(x) := 0\ \forall x \in X\ ,\ i \in N$;
2: $V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N,\ o_i \in O_i\ ,\ \bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
3: **compute** X_i^*;
4: **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 $G(V)$
1: **if** $|X_i^*(V) \backslash O_i| > 0$ **then**
2: $s :=$ any strategy in $X_i^*(V) \backslash O_i$; $lev_{best} := 0$;
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: $lev := ExactLev(V + W, i)$;
7: **if** $lev > lev_{best}$ **then**
8: $lev_{best} := lev$;
9: **for all** $o_{-i} \in O_{-i}$ **do**
10: $W_i(o_i, o_{-i}) := 0$;
11: **return** lev_{best};
12: **if** $i > 1$ **return** $ExactLev(V, i-1)$;
13: **else return** $\min_{o \in O}\{U(o) - V(o)\}$;

In the following we will provide an algorithm which computes a convex 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 [5] in order to compute a set's exact cost. In order to compute $LEV(O)$, we can apply Algorithm 2 for all convex subsets and return the largest value found.

Theorem 6. *Algorithm 2 computes a strategy profile set's exact leverage in time* $\mathrm{O}\big(|X|^2 \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i| - 1}) + n|O| \max_{i \in N}(|O_i|^{n|X_i^* \backslash O_i|})\big)$.

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

[3] Note that we do not provide algorithms for computing the malicious leverage but for the benevolent leverage only. However, it is straightforward to adapt our algorithms for the benevolent leverage.

number of players (compare the analysis of Algorithm 1 in [5]). □

4 Uniform Implementation Cost

We will now turn our attention to the situation of less anxious mechanism designers who anticipate uniform rather than worst-case implementation cost. They assume the players to select one of their non-dominated strategies uniformly at random. This is a reasonable assumption if the players do only know their own utilities.

Note that for strategy profile sets O with $k^*_{UNI}(O) = 0$ any exact implementation V must have zero payments for any profile inside O, i.e. $V(o) = 0\ \forall o \in O$. Thus, for 0-implementable strategy profile sets, the concepts of worst case exact cost and uniform exact cost coincide, i.e., $k^*_{UNI}(O) = 0$ iff $k^*(O) = 0$. Therefore, Algorithm 2 from [5] decides if O has uniform exact cost of 0 for the uniform case in polynomial as well.

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

Theorem 7. *In games with at least two (three) players, the problem of finding a strategy profile set's exact (non-exact) uniform implementation cost is* **NP***-hard.*

Proof. *Exact Case:* 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 $S_i \subset \mathcal{U}$, SETCOVER 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 SETCOVER 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 SETCOVER 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.[4] 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 $\varnothing_{o \in O} V(o) = m/|O|$. 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 larger costs. 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.

	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. 3. Payoff matrix for Player 1 in a game which reduces the SETCOVER 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.

Non-Exact Case: We give a similar reduction of SETCOVER 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 SETCOVER problem instance $SC = (\mathcal{U}, \mathcal{S})$, we can construct a game $G = (N, X, U)$ where $N = \{1, 2, 3\}$, $X_1 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m\}$, $X_2 = \{e_1, e_2, \ldots, e_l, s_1, s_2, \ldots, s_m, d, r\}$, $X_3 = \{a, b\}$. 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

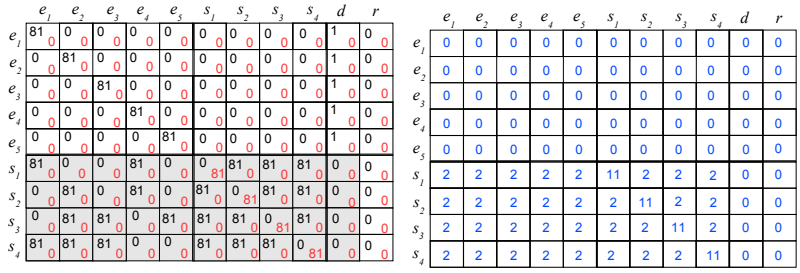

Fig. 4. Payoff matrix for Player 1 and Player 2 given Player 3 chooses a and payoff matrix for Player 3 when she plays strategy b in a game which reduces a SETCOVER instance $SC = (\mathcal{U}, \mathcal{S})$ where $\mathcal{U} = \{e_1, e_2, e_3, e_4, e_5\}$, $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, $S_1 = \{e_1, e_4\}, S_2 = \{e_2, e_4\}, S_3 = \{e_2, e_3, e_5\}, S_4 = \{e_1, e_2, e_3\}$ 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 SETCOVER problem.

$\{(x_1, x_2, x_3) | x_1 = s_i \wedge (x_2 = e_i \vee x_2 = s_i \vee x_2 = d) \wedge (x_3 = a)\}$. See Fig. 4 for an example.

First, note the fact that any implementation of O will 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.[5] 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 SETCOVER 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 $cost_{s_i} = (3(l+m)+1)/(l+m+1)$. A non-selected row s_j contributes $cost_{s_j} = (3(l+m))/(l+m+1) < cost_{s_i}$. 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 $\varnothing_{i=1}^{m} cost_{s_i} = (3(l+m)+1)/(l+m+1) < 3$.

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 costs are strictly larger. Assume a set yielding lower cost implements α strategies for Player 1, β strategies e_i and γ strategies s_j for

[4] 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 payoff $V_1(s_j, r)$ in the extra column r.

[5] 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.

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 costs 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 costs are 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 costs amount 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 SETCOVER 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 SETCOVER [2,6] carry over to our problem.

Theorem 8. *No polynomial-time algorithm can achieve an approximation ratio better than* $\Omega(n \max_i\{\log|X_i^* \setminus O_i|\})$ *for both the exact and non-exact implementation costs within any function of the input length unless* **P**=**NP**.

Proof. *Exact Case:* In order to prove this claim, a reduction similar to the one in the proof of Theorem 7 can be applied. Consider again a SETCOVER 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 SETCOVER instance has to be solved for each Player i $\forall i \in \{1, \ldots, n-1\}$. According to the well-known inapproximability results for SETCOVER, no polynomial time algorithm exists which achieves a better approximation ratio than $\Omega(\log|X_i^* \setminus O_i|)$ for each Player i, unless $\mathbf{P} = \mathbf{NP}$, and the claim follows.

Non-Exact Case: We use the inapproximability results for SETCOVER 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 7 for the non-exact case, independently of all other players' payoffs. Hence, SETCOVER has to be solved for $n/3$ players. □

5 Uniform Leverage

A mechanism designer calculating her average case cost is more optimistic than an anxious designer. 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 9. *A set's uniform (malicious) leverage is always larger or equal the set's (malicious) leverage.*

PROOF. $lev_{UNI}(O) = \max_{V \in \mathcal{V}(O)}\{\varnothing_{z \in X^*(V)}\{U(z) - V(z)\}\} - \varnothing_{x^* \in X^*(V)} U(x^*)$ $\geq \max_{V \in \mathcal{V}(O)}\{\min_{z \in X^*(V)}\{U(z) - V(z)\}\} - \max_{x^* \in X^*(V)} U(x^*) = lev(O)$, and $mlev_{UNI}(O) = \varnothing_{x^* \in X^*(V)} U(x^*) - \min_{V \in \mathcal{V}(O)}\{\varnothing_{z \in X^*(V)}\{U(z) + 2V(z)\}\}$ $\geq \min_{x^* \in X^*(V)}\{U(x^*)\} - \min_{V \in \mathcal{V}(O)}\{\max_{z \in X^*(V)}\{U(z) + 2V(z)\}\} = mlev(O)$. □

Another difference concerns the sets O intersecting with X^*, i.e., $O \cap X^* \neq \emptyset$: Unlike the worst-case leverage (Theorem 4), the uniform leverage can exceed zero in these cases, as can be verified by calculating O's leverage in Fig. 3.

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 SETCOVER problem in Theorem 7, and by using a lower bound for the approximation quality of the SETCOVER 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 **P**=**NP**, there exists no polynomial time algorithm approximating the leverage of a game within any function of the input length.

Theorem 10. *For games with at least two (three) players, the problem of computing a strategy profile set's exact (non-exact) uniform (malicious) leverage is* **NP***-hard. Furthermore, the (exact) uniform leverage of O cannot be approximated in polynomial time within any function of the input length unless* **P**=**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 7). Due to the fact that any $z \in O$ is also in $X^*(V)$ for any $V \in \mathcal{V}^*(O)$, $lev_{UNI}(O) = \max_{V \in \mathcal{V}^*(O)}\{\varnothing_{z \in X^*(V)}\{U(z) -$

$V(z)\}\} - \varnothing_{z\in X^*} U(x^*) = \max_{V\in\mathcal{V}^*(O)}\{\varnothing_{z\in X^*(V)} U(z) - \varnothing_{z\in X^*(V)} V(z)\} - \varnothing_{x^*\in X^*} U(x^*) = \varnothing_{z\in X^*(V)} U(z) - \min_{V\in\mathcal{V}^*(O)}\{\varnothing_{z\in X^*(V)} V(z)\} - \varnothing_{x^*\in X^*} U(x^*) = \varnothing_{z\in X^*(V)} U(z) - \mathbf{k^*_{UNI}}(\mathbf{O}) - \varnothing_{x^*\in X^*} U(x^*)$.

$mlev_{UNI}(O) = \varnothing_{x^*\in X^*} U(x^*) - \min_{V\in\mathcal{V}^*(O)}\{\varnothing_{z\in X^*(V)}\{U(z) + 2V(z)\}\} = \varnothing_{x^*\in X^*} U(x^*) - \varnothing_{z\in X^*(V)} U(z) - 2\min_{V\in\mathcal{V}^*(O)}\{\varnothing_{z\in X^*(V)} V(z)\} = \varnothing_{x^*\in X^*} U(x^*) - \varnothing_{z\in X^*(V)} U(z) - 2\mathbf{k^*_{UNI}}(\mathbf{O})$.

Observe that $\varnothing_{x^*\in X^*} U(x^*)$ and $\varnothing_{z\in X^*(V)} U(z)$ can be computed easily. Moreover, as illustrated in the proof of Theorem 5, 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 7 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 $k_{UNI}(O) = (\varnothing_{x^*\in X^*} U(x^*) - MLEV_{UNI}(O))/2$. 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 SETCOVER problem in Theorem 7 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 SETCOVER instance, $\epsilon > 0$, and $\mathcal{LB}$ denotes a lower bound for the approximation quality of the SETCOVER problem. Observe that X^* consists of all strategy profiles of column r. The target set we want to implement exactly is given by $O_1 = \{s_1, ..., s_m\}$ and $O_2 = \{e_1, ..., e_l, d\}$. We compute $lev^{opt}_{UNI} = \varnothing_{o\in O} U(o) - \varnothing_{x\in X^*} U(x) - k = \sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - \sum_{i=1}^{m} |S_i|(m+1)/(ml+m) - (-k\mathcal{LB} - \epsilon) - k = k(\mathcal{LB} - 1) + \epsilon$.

As no polynomial time algorithm can approximate k within a factor $\mathcal{LB}$, $LEV^{approx}_{UNI} \leq \epsilon$. Since $\lim_{\epsilon\to 0} LEV^{opt}_{UNI}/LEV^{approx}_{UNI} = \infty$ the claim follows for a benevolent mechanism designer.

For malicious mechanism designers, we modify the utilities of the game from the proof of Theorem 8 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 7'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 SETCOVER instance, $\epsilon > 0$, and

$\mathcal{LB}$ denotes a lower bound for the approximation quality of the SETCOVER 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 7 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. □

Algorithms. To round off our analysis of the uniform (malicious) leverage, we investigate algorithms for risk neutral mechanism designers. Recall Algorithm 1 in Section 3 which computes the leverage of singletons of a desired strategy profile set. We can adapt this algorithm in Line 3 and 4 to accommodate the definition of the uniform leverage, i.e., set $mlev[x] := \varnothing_{x^* \in X^*} U(x^*)$ and $mlev[x] := -mlev[x]$. This algorithm thus helps to find an optimal singleton.

A benevolent mechanism designer can adapt Algorithm 2 in order to compute $LEV^*_{UNI}(O)$: She only has to change Line 4 to $\max\{0, ExactLev(V, n) - \varnothing_{x^* \in X^*} U(x^*)\}$ and 'min' in Line 13 to '$\varnothing$'. Invoking this algorithm for any $O' \subseteq O$ yields the subset O with maximal leverage $LEV_{UNI}(O)$.

Due to Theorem 10 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, as for the worst case leverage.

6 Conclusion

This paper has studied benevolent and malicious mechanism designers whose goal is to change the game's outcome if the improvement or deterioration in social welfare exceeds the cost. We have presented several algorithms and computational complexity results for cautious and optimistic mechanism designers. Our models still pose many interesting questions which have to be tackled in 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

1. A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR Fault Tolerance for Cooperative Services. In *Proc. 20th ACM Symposium on Operating Systems Principles (SOSP)*, 2005.
2. N. Alon, D. Moshkovitz, and M. Safra. Algorithmic Construction of Sets for k-Restrictions. *ACM Transactions on Algorithms (TALG)*, pages 153–177, 2006.
3. M. Babaioff, M. Feldman, and N. Nisan. Combinatorial Agency. In *Proc. 7th ACM Conference on Electronic Commerce (EC)*, pages 18–28, 2006.

4. M. Babaioff, R. Kleinberg, and C. Papadimitriou. Congestion Games with Malicious Players. In *Proc. ACM Conference on Electronic Commerce (EC), San Diego, CA, USA*, 2007.
5. R. Eidenbenz, Y. A. Oswald, S. Schmid, and R. Wattenhofer. Mechanism Design by Creditability. In *Proc. 1st International Conference on Combinatorial Optimization and Applications (COCOA), Springer LNCS 4616*, 2007.
6. U. Feige. A Threshold of $\log n$ for Approximating Set Cover. *Journal of the ACM*, pages 634–652, 1998.
7. J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *Proc. 21st Annual ACM Symp. on Principles of Distributed Computing (PODC)*, 2002.
8. D. Monderer and M. Tennenholtz. k-Implementation. In *Proc. 4th ACM Conference on Electronic Commerce (EC)*, pages 19–28, 2003.
9. T. Moscibroda, S. Schmid, and R. Wattenhofer. When Selfish Meets Evil: Byzantine Players in a Virus Inoculation Game. In *Proc. 25th Annual Symposium on Principles of Distributed Computing (PODC), Denver, Colorado, USA*, 2006.
10. N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, 1999.
11. Osborne and Rubinstein. *A Course in Game Theory.* MIT Press, 1994.
12. T. Roughgarden. Stackelberg Scheduling Strategies. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.
13. I. Segal. Contracting with Externalities. *The Quarterly Journal of Economics*, pages 337–388, 1999.