Mechanism Design by Creditability*

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

Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland

Abstract. This paper attends to the problem of a mechanism designer seeking to influence the outcome of a strategic game based on her creditability. The mechanism designer offers additional payments to the players depending on their mutual choice of strategies in order to steer them to certain decisions. Of course, the mechanism designer aims at spending as little as possible and yet implementing her desired outcome. We present several algorithms for this optimization problem both for singleton target strategy profiles and target strategy profile regions. Furthermore, the paper shows how a bankrupt mechanism designer can decide efficiently whether strategy profiles can be implemented at no cost at all. Finally, risk-averse players and dynamic games are examined.

1 Introduction

The quest for a deeper understanding of our world and its highly interconnected systems and processes often requires a huge amount of computational resources which can only be obtained by connecting thousands of computers. Similarly to agents in socio-economic systems, the computers in such networks often operate on a decentralized control regime, and represent different stake-holders with different objectives. Therefore, in addition to mere technical challenges, a system designer often has to take into account sociological and economic aspects as well when reasoning about protocols for maximizing system performance.

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 often suffer from inefficiency due to effects of selfishness. The concept of the *price of anarchy* allows 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, a mechanism designer cannot change the rules of interactions. However, she may be able to influence the agents' behavior by offering payments for certain outcomes. On this account, we consider a mechanism designer whose

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

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 the performance of a given system merely by creditability, i.e., without any payments at all.

This paper presents several results for this problem. The first correct algorithm for finding an exact, incentive compatible implementation of a desired set of outcomes is given. We also show how a bankrupt mechanism designer can decide in polynomial time if a set of outcomes can be implemented at no costs at all, and an interesting connection to best response graphs is established. We propose and analyze efficient heuristic algorithms and demonstrate their performance. Furthermore, we extend our analysis for risk-averse behavior and study dynamic games where the mechanism designer offers payments in each round.

The remainder of this paper is organized as follows. Section 2 reviews related work. Our model and some basic game theory definitions are introduced in Section 3. In Section 4, algorithms for computing exact and non-exact implementations are proposed. Section 5 presents simulation results. Risk-averse players and dynamic games are studied in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

The mathematical tools of game theory have become popular in computer science recently as they allow to gain deeper insights into the socio-economic complexity of today's distributed systems. Game theory combines algorithmic ideas with concepts and techniques from mathematical economics. Popular problems in computer science studied from a game theoretic point of view include *virus propagation* [1], *congestion* [3], *network creation* [7], among many others.

The observation that systems often perform poorly in the presence of selfish players has sparked research for countermeasures [6,10]. For example, Cole et al. [4,5] 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.

In [11], Monderer and Tennenholtz consider an interested third party who attempts to lead selfish players to act in a desired way. The third party can neither enforce behavior nor change the system, she can only influence the game's outcome by announcing non-negative monetary transfers conditioned on the behavior of the agents. The authors show that the interested third party might be able to induce a desired outcome at very low costs. In particular, they prove that any pure Nash equilibrium of a game with complete information has a *zero-implementation*, i.e., it can be transformed into a dominant strategy profile at zero cost. 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 cost.

This paper extends [11] in various respects. Several new algorithms are provided, for instance a polynomial time algorithm for deciding whether a set of strategy profiles has a 0-implementation. In addition, we suggest polynomial-time heuristic algorithms and simulate their performance. Connections to graph-theoretic concepts are pointed out and we generalize the theorem by Monderer and Tennenholtz on the cost of Nash equilibria. Their algorithm for computing an optimal exact implementation is corrected, and we provide evidence that their $\mathbf{NP}$-hardness proof of deciding whether a k-implementation exists is wrong. Furthermore, the concept of implementation is generalized for other game theoretic models. We examine players aiming at maximizing the average payoff and show how the mechanism designer can find such implementations. As another contribution, this paper considers the case of risk-averse players as well as the resulting complexity of computing the optimal implementation cost, and initiates the study of mechanism design by creditability in round based dynamic games.

Our work is also related to *Stackelberg theory* [12] where a fraction of the entire population is orchestrated by a global leader. In contrast to our paper, the leader is not bound to offer any incentives to follow her objectives. Finally, in the recent research thread of *combinatorial agencies* [2], 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.

3 Model

Game Theory A *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 $X := X_1 \times X_2 \times \ldots \times X_n$. 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 of x_i. Finally, $U = (U_1, U_2, \ldots, U_n)$ 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. 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. The set of *best responses* $B_i(x_{-i})$ for Player i given the other players' actions is defined as $B_i(x_{-i}) := \{x_i | U_i(x_i, x_{-i}) = \max_{x_j \in X_i \backslash \{x_i\}} U_i(x_j, x_{-i})\}$. A *Nash equilibrium* is a strategy profile $x \in X$ such that for all $i \in N$, $x_i \in B_i(x_{-i})$.

Mechanism Design by Creditability This paper acts on the classic assumption that players are rational and always choose a non-dominated strategy. Additionally, it is

assumed that players 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. 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 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 $X^*(V)$. 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 $O_i := \{x_i | \exists x_{-i} \in X_{-i} \text{ s.t. } (x_i, x_{-i}) \in O\}$ 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. For a desired strategy profile region O, we say that payments V *implement* O if $\emptyset \subset X^*(V) \subseteq O$. V is called a *k-implementation* if, in addition $\sum_{i=1}^{n} V_i(x) \leq k$, $\forall x \in X^*(V)$. 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 $\sum_{i=1}^{n} V_i(x) \leq k \ \forall x \in X^*(V)$. 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 $\max_{x \in X^*(V)} \sum_{i=1}^{n} V_i(x) = k(O)$. 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 region $O = \{z\}$ 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 $\{z\}$ and $V(z)$ instead of $\sum_{i \in N} V_i(z)$. Observe that only subsets of X which are in $2^{X_1} \times 2^{X_2} \times \ldots \times 2^{X_n} \subset 2^{X_1 \times X_2 \times \ldots \times X_n}$ can be implemented exactly. We call such a subset of X a *convex strategy profile region*.[1]

4 Algorithms and Analysis

4.1 Exact Implementation

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

[1] These regions 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.

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 $M_i^G : X_i \backslash X_i^* \rightarrow X_i^*$, 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. 1 for an example. When imple-

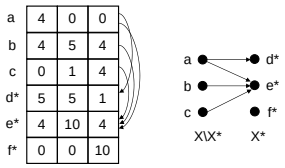

Fig. 1. Game from a single player's point of view with corresponding relation of dominated ($X_i \backslash X_i^* = \{a, b, c\}$) to non-dominated strategies ($X_i^* = \{d^*, e^*, f^*\}$).

menting a strategy profile region O exactly, the mechanism designer creates a modified game $G(V)$ with a new relation $M_i^V : X_i \setminus O_i \rightarrow O_i$ 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 $(\bar{o}_i, x_{-i})$ where $\bar{o}_i \in X_i \backslash O_i,\ x_{-i} \in X_{-i}$. 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 $S_i(o_i)$:=$\{s \in X_i \backslash O_i | M_i^V(s) = o_i\}$ 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 $V_i(o_i, o_{-i}) = \max\left\{0, \max_{s \in S_i(o_i)}(U_i(s, o_{-i}))\right\} - U_i(o_i, o_{-i})$ 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 $M_i^V : X_i \backslash O_i \rightarrow O_i$.

Our algorithm $\mathcal{ALG}_{exact}$ (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.

Theorem 1. *$\mathcal{ALG}_{exact}$ computes a strategy profile region's optimal exact implementation cost 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. $\mathcal{ALG}_{exact}$ is correct as it checks all possible relations in the relation set $M^V = \{M_i^V : X_i^*(V) \backslash O_i \rightarrow O_i\ \forall i \in N\}$ recursively by calling the subroutine $ExactK$ in

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

Input: Game G, convex 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: $V_i(o_i, \bar{o}_{-i}) := \infty\ \forall i \in N, o_i \in O_i\ , \bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
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** $|X_i^*(V) \backslash O_i| > 0$ **then**
2: $s :=$ any strategy in $X_i^*(V) \backslash O_i$; $k_{best} := \infty$;
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** $\max_{o \in O} \sum_i V_i(o)$;

Line 6. Therefore, it must find the relation set which corresponds to an implementation with optimal cost.

It remains to prove the algorithm's runtime. Computing the non-dominated region X^* by checking for each strategy whether it is dominated takes time $\sum_{i=1}^{n} \binom{|X_i|}{2} |X_{-i}|$ $= O(n|X|^2)$. The complexity of this computation asymptotically dominates the runtime required by Lines 1 and 2. We next examine the complexity of subroutine *ExactK*. Computing Line 1 costs $|X|^2$, the two for-loops in Lines 3 and 4 are executed $|O|$ times, and *ExactK* is called $|O_i|$ times (Line 6). Hence, we derive the following (asymptotic) recursive equations for the runtime $T_i(\ell)$ for *ExactK*(V, i) if i has yet ℓ strategies to dominate:

$$T_i(\ell) = \begin{cases} |X|^2 + |O| + |O_i| T_i(\ell - 1) & \text{if } (0 < \ell < |X_i^* \setminus O_i|) \wedge (i \in N) \\ T_{i-1}(|X_{i-1}^* \setminus O_{i-1}|) & \text{if } \ell = 0 \wedge i \in N \\ n|O| & \text{if } \ell = 0 \wedge i = 0 \end{cases}$$

For $\ell_i = |X_i^* \setminus O_i|$, we obtain $T_i(\ell_i) = |O_i|^{\ell_i - 1}|X|^2 + |O_i|^{\ell_i} T_{i-1}(\ell_{i-1})$ if $i > 1$. Let $a_i = |O_i|^{\ell_i - 1}|X|^2, b_i = |O_i|^{\ell_i}$ and $a = \max_{i \in N} a_i, b = \max_{i \in N} b_i$; hence

$$\begin{aligned} T_i(\ell_i) &= a_i + b_i T_{i-1}(\ell_{i-1}) \\ &= a \left[\sum_{j=1}^{i} \prod_{k=1}^{j-1} b_k \right] + \left[\prod_{k=1}^{i} b_k \right] T_1(0) \\ &= a \sum_{j=1}^{i} b^{j-1} + b^i n|O| \\ &= a b^{i-1} + b^i n|O| \end{aligned}$$

and the claim follows. □

Note that $\mathcal{ALG}_{exact}$ has a large time complexity. In fact, a faster algorithm for this problem, called *Optimal Perturbation Algorithm* has been presented in [11]. 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 an agent's payoffs in the original game. The algorithm terminates as soon as all strategies in $X_i^* \setminus O_i$ are dominated. Unfortunately, this algorithm does not always return an optimal implementation. Sometimes, as we will show in Appendix A, the optimal perturbation algorithm increases the values unnecessarily. In fact, we even conjecture that deciding whether an k-exact implementation exists is **NP**-hard.

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

Bankrupt Mechanism Designers Imagine a mechanism designer who is broke. At first sight, it seems that without any money, she will hardly be able to influence the outcome of a game. However, this intuition ignores the power of creditability: a game can have 0-implementable regions.

Let V be an exact implementation of O with exact costs $k^*(O)$. It holds that if $k^*(O) = 0$, V cannot contain any payments larger than 0 in O. Consequently, for an region O to be 0-implementable exactly, any strategy s outside O_i must be dominated within the range of O_{-i} by a o_i, or there must be one o_i for which no payoff $U_i(s, o_{-i})$ is larger than $U_i(o_i, o_{-i})$. In the latter case, the strategy o_i can still dominate s by using a payment $V(o_i, x_{-i})$ with $x_{-i} \in X_{-i} \setminus O_{-i}$ outside O. Note that this is only possible under the assumption that $O_{-i} \subset X_{-i}\ \forall i \in N$.

$\mathcal{ALG}_{bankrupt}$ (cf. Algorithm 2) describes how a bankrupt designer can decide in polynomial time whether a certain region is 0-implementable. It proceeds by checking for each Player i if the strategies in $X_i^* \setminus O_i$ are dominated or "almost" dominated within the range of O_{-i} by at least one strategy inside O_i. If there is one strategy without such a dominating strategy, O is not 0-implementable exactly. On the other hand, if for every strategy $s \in X_i^* \setminus O_i$ such a dominating strategy is found, O can be implemented exactly without expenses.

Algorithm 2 Exact 0-Implementation ($\mathcal{ALG}_{bankrupt}$)

Input: Game G, convex region O with $O_{-i} \subset X_{-i}\ \forall i$
Output: $\top$ if $k^*(O) = 0$, $\bot$ otherwise

```
1: compute X*;
2: for all i ∈ N do
3:    for all s ∈ X_i^* \ O_i do
4:       dZero := ⊥;
5:       for all o_i ∈ O_i do
6:          b := ⊤;
7:          for all o_{-i} ∈ O_{-i} do
8:             b := b ∧ (U_i(s, o_{-i}) ≤ U_i(o_i, o_{-i}));
9:          dZero := dZero ∨ b;
10:      if ¬ dZero then
11:         return ⊥;
12: return ⊤;
```

Theorem 2. *Given a convex strategy profile region O where $O_{-i} \subset X_{-i}\ \forall i$, Algorithm $\mathcal{ALG}_{bankrupt}$ decides whether O has an exact 0-implementation in time* $\mathrm{O}\left(n\left|X\right|^2\right)$.

PROOF. $\mathcal{ALG}_{bankrupt}$ is correct because it checks for each yet to be dominated strategy $s \in X_i^* \backslash O_i$ whether it can be dominated by one $o_i \in O_i$ at zero cost. This is the property that makes O exactly 0-implementable. Computing X^* takes time $\mathrm{O}\left(n\left|X\right|^2\right)$. All other costs are asymptotically negligible. □

Best Response Graphs Best response strategies maximize the payoff for a player given the other players' decisions. For now, let us restrict our analysis to games where the sets of best response strategies consist of only one strategy for each $x_{-i}\ \forall i \in N$. Given a game G, we construct a directed *best response graph* $\mathcal{G}_G$ with vertices v_x for strategy profiles $x \in X$ iff x is a best response for at least one player, i.e., if $\exists i \in N$ such that $x_i \in B_i(x_{-i})$. There is a directed edge $e = (v_x, v_y)$ iff $\exists i \in N$ such that $x_{-i} = y_{-i}$ and $\{y_i\} = B_i(y_{-i})$. In other words, an edge from v_x to v_y, indicates that it is better to play y_i instead of x_i for a player if for the other players' strategies $x_{-i} = y_{-i}$. A strategy profile region $O \subset X$ has a *corresponding subgraph* $\mathcal{G}_{G,O}$ containing the vertices $\{v_x | x \in O\}$ and the edges which both start and end in a vertex of the subgraph. We say $\mathcal{G}_{G,O}$ has an *outgoing edge* $e = (v_x, v_y)$ if $x \in O$ and $y \notin O$. Note that outgoing edges are not in the edge set of $\mathcal{G}_{G,O}$. Clearly, it holds that if a singleton x's corresponding subgraph $\mathcal{G}_{G,\{x\}}$ has no outgoing edges then x is a *Nash equilibrium*. More generally, we make the following observation.

Theorem 3. *Let G be a game and $|B_i(x_{-i})| = 1\ \forall i \in N, x_{-i} \in X_{-i}$. If a convex region O has an exact 0-implementation, then the corresponding subgraph $\mathcal{G}_{G,O}$ in the game's best response graph has no outgoing edges.*

PROOF. Let V be an exact 0-implementation of O. Note that $V(o) = 0\ \forall o \in O$, otherwise the cost induced by V are larger than 0. Assume for the sake of contradiction that

$\mathcal{G}_{G,O}$ has an outgoing edge. Let $x \in O$ be a strategy profile for which its corresponding vertex v_x has an outgoing edge e to $v_y, y \in X \backslash O$. Since $V(x)$ is 0, $\mathcal{G}_{G(V),O}$ still has the same outgoing edge e. This means that for one Player j it is better to play strategy y_j in $G(V)$ than to play x_j given that $x_{-j} = y_{-j}$. Since y_j is not dominated by any strategy in O_j, Player j will hence choose also strategies outside O_j and therefore V is not a correct implementation of O, thus contradicting our assumption. □

In order to extend best response graphs to games with multiple best responses, we modify the edge construction as follows: In the general best response graph $\mathcal{G}_G$ of a game G there is a directed edge $e = (v_x, v_y)$ iff $\exists i \in N$ s.t. $x_{-i} = y_{-i}$, $y_i \in B_i(y_{-i})$ and $|B_i(y_{-i})| = 1$.

Corollary 1. *Theorem 3 holds for arbitrary games.*

Note that Theorem 3 is a generalization of Monderer and Tennenholtz' Corollary 1 in [11]. They discovered that for a singleton x, it holds that x has a 0-implementation if and only if x is a Nash equilibrium. While their observation covers the special case of singleton-regions, our theorem holds for any strategy profile region. Unfortunately, for general regions, one direction of the equivalence holding for singletons does not hold anymore due to the fact that 0-implementable regions O must contain a player's best response to any o_{-i} but they need not contain best responses exclusively.

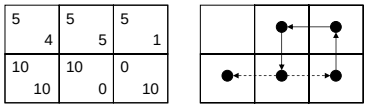

Fig. 2. Sample game G with best response graph $\mathcal{G}_G$. The Nash equilibrium in the bottom left corner has no outgoing edges. The dotted arrows do not belong to the edge set of $\mathcal{G}_G$ as the row has multiple best responses.

4.2 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 costs, we could find $k(O)$ by computing $k^*(O')$. Apart from the fact that finding an optimal implementation includes solving the – believed to be $\mathbf{NP}$-hard – optimal exact implementation cost problem for at least one subregion of O, finding this subregion might also be $\mathbf{NP}$-hard since there are exponentially many possible subregions. In fact, a reduction from the SAT problem is presented in [11]. 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 $\mathbf{NP}$-hard, i.e., we assume that no algorithm can do much better than performing a brute force computation of the exact implementation costs (cf. Algorithm 1) of all possible subsets, unless $\mathbf{NP} = \mathbf{P}$.

Conjecture 2. Finding an optimal implementation of a strategy region is $\mathbf{NP}$-hard.

For the special case of zero cost regions, Theorem 3 implies the following result.

Corollary 2. *If a strategy profile region O has zero implementation cost then the corresponding subgraph $\mathcal{G}_{G,O}$ in the game's best response graph contains a subgraph $\mathcal{G}_{G,O'}, O' \subseteq O$, with no outgoing edges.*

Corollary 2 is useful to a bankrupt mechanism designer since searching the game's best response graph for subgraphs without outgoing edges helps her spot candidates for regions which can be implemented by mere creditability. In general though, the fact that finding optimal implementations seems computationally hard raises the question whether there are polynomial time algorithms achieving good approximations. As mentioned in Section 4.1, each V implementing a region O defines a domination relation $M_i^V : X_i \setminus O_i \rightarrow O_i$. This observation leads to the idea of designing heuristic algorithms that find a correct implementation by establishing a corresponding relation set $\{M_1, M_2, \ldots, M_n\}, M_i : X_i^* \backslash O_i \rightarrow O_i$ where each $x_i^* \in X_i^* \backslash O_i$ maps to at least one $o_i \in O_i$. These algorithms are guaranteed to find a correct implementation of O, however, the corresponding implementations may not be cost-optimal.

Our greedy algorithm $\mathcal{ALG}_{greedy}$ (cf. Algorithm 3) associates each strategy x_i^* yet to be dominated with the o_i with minimal distance Δ_G to x_i^*, i.e., the maximum value that has to be added to $U_i(x_i', x_{-i})$ such that x_i' dominates x_i: $\Delta_G(x_i, x_i') := \max_{x_{-i} \in X_{-i}} \max(0, U_i(x_i, x_{-i}) - U_i(x_i', x_{-i}))$. Similarly to the greedy approximation algorithm for the *set cover problem* [8,9] which chooses in each step the subset covering the most elements not covered already, $\mathcal{ALG}_{greedy}$ selects a pair of (x_i^*,o_i) such that by dominating x_i^* with o_i, the number of strategies in $X_i^* \backslash O_i$ that will be dominated therewith is maximal. Thus, in each step there will be an o_i assigned to dominate x_i^* which has minimal dominating cost. Additionally, $\mathcal{ALG}_{greedy}$ takes any opportunity to dominate multiple strategies. $\mathcal{ALG}_{greedy}$ is described in detail in Algorithm 3. It returns an implementation V of O; to determine V's cost, one needs to compute $\max_{x^* \in X^*(V)} \sum_{i \in N} V_i(x^*)$.

Theorem 4. *$\mathcal{ALG}_{greedy}$ returns an implementation of a convex strategy profile region $O \in X$ in time* $\mathrm{O}\big(n|X|^2 \max_{i \in N} |X_i^* \setminus O_i| + n|X| \max_{i \in N} |X_i^* \setminus O_i|^3\big)$.

PROOF. $\mathcal{ALG}_{greedy}$ terminates since in every iteration of the while-loop, there is at least one newly dominated strategy. The payment matrix V returned is an implementation of O because the while-condition $X_i^*(V) \nsubseteq O_i$ turned false for all $i \in N$ and thus, it holds that $X_i^*(V) \subseteq O_i \; \forall i \in N$.

Line 1 takes time $\mathrm{O}(|X|\, n)$. Asymptotically, computing X^* costs $\mathrm{O}\big(|X|^2 n\big)$. Setting the payments $V_i(o_i, \bar{o}_{-i})$ to infinity (Line 4) for all players takes time $\mathrm{O}(n\,|X \setminus O|)$.

The while-loop (Lines 5-18) is executed $|X_i^* \setminus O_i|$ times, and evaluating the while-loop's condition takes at most time $|X^2|$. One iteration of the while-loop takes time

$$\underbrace{|X_i^* \setminus O_i|}_{Line\ 7} \cdot (\underbrace{|O_i||X_{-i}|}_{Line\ 8} + \underbrace{|O_{-i}|}_{Line\ 9} + \underbrace{|X_i^* \setminus O_i|}_{Line\ 12} \underbrace{(|X| + |O_{-i}|)}_{Line\ 13}) + \underbrace{O_{-i}}_{Line 17}$$

Combining the above expressions yields the claim. □

$\mathcal{ALG}_{red}$ (cf. Algorithm 4) is a more sophisticated algorithm applying $\mathcal{ALG}_{greedy}$. Instead of terminating when the payment matrix V implements O, this algorithm continues to search for a payment matrix inducing even less cost. It uses $\mathcal{ALG}_{greedy}$ to approximate the cost repeatedly, varying the region to be implemented. As $\mathcal{ALG}_{greedy}$ leaves the while-loop if $X_i^*(V) \subseteq O_i$, it might miss out on cheap implementations where $X_i^*(V) \subseteq Q_i,\ Q_i \subset O_i$. $\mathcal{ALG}_{red}$ examines some of these subsets as well by calling $\mathcal{ALG}_{greedy}$ for some Q_i. If we manage to reduce the cost, we continue with $O_i := Q_i$ until neither the cost can be reduced anymore nor any strategies can be deleted from any O_i.

Theorem 5. *Let T_g denote the runtime of $\mathcal{ALG}_{greedy}$. $\mathcal{ALG}_{red}$ returns an implementation of O in time* $\mathrm{O}(n|O| \max_{i \in N} |O_i|(|O| + n + T_g))$.

PROOF. $\mathcal{ALG}_{red}$ terminates because the condition of the while-loop does not hold anymore if there are no more strategies left, and because in at most every $\max_{i \in N} |O_i|^{th}$ iteration at least one strategy is removed. Clearly, the while-loop is repeated at most $\max_{i \in N} |O_i| \cdot |O|$ times, as a removed strategy is never added again. We iterate over all players (time n) and look for a strategy to be removed (time $|O|$ in Line 5). Evaluating the if-clause (Line 6) requires time $n + |O_i|$. Finally in Line 10, the greedy algorithm is called recursively, which is assumed to take time T_g. The time complexity of all other operations can be neglected. □

An alternative heuristic algorithm for computing a region O's implementation cost retrieves the region's cheapest singleton, i.e., $\min_{o \in O} k(o)$, where a singleton's implementation cost is $k(o) = \min_{o \in O} \sum_{i \in N} \max_{x_i \in X_i} (U_i(x_i, o_{-i}) - U_i(o_i, o_{-i}))$ [11]. The best singleton heuristic algorithm performs quite well for randomly generated games as our simulations reveal (cf. Section 5), but it can result in an arbitrarily large k in the worst case: Fig. 3 depicts a game where 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 raises the following question: What characteristics are stringent for a game and a corresponding desired strategy profile region O such that only non-singleton subregions bear the optimal implementation cost? Clearly, we have to consider games where at least one player has four or more strategies, at least two of which must not be in O_i. Moreover, it must cost less to dominate them with two strategies in O_i than with just one strategy in O_i.

5 Simulation

All our algorithms return correct implementations of the desired strategy profile sets and – apart from the recursive algorithm $\mathcal{ALG}_{exact}$ for the optimal exact implementation –

Algorithm 3 Greedy Algorithm $\mathcal{ALG}_{greedy}$

Input: Game G, convex target region O
Output: Implementation V of O

$V_i(x) := 0; W_i(x) := 0 \; \forall x \in X$, $i \in N$;
compute X^*;
for all $i \in N$ **do**
  $V_i(o_i, \bar{o}_{-i}) := \infty \; \forall o_i \in O_i$, $\bar{o}_{-i} \in X_{-i} \backslash O_{-i}$;
  while $X_i^*(V) \not\subseteq O_i$ **do**
    $c_{best} := 0; m_{best}$:=null; s_{best} :=null;
    for all $s \in X_i^*(V) \backslash O_i$ **do**
      $m := \arg\min_{o_i \in O_i}(\Delta_{G(V)}(s, o_i))$;
      for all $o_{-i} \in O_{-i}$ **do**
        $W_i(m, o_{-i})$:=max$(0, U_i(s, o_{-i}) - (U_i(m, o_{-i}) + V_i(m, o_{-i})))$;
      $c := 0$;
      for all $x \in X_i^* \setminus O_i$ **do**
        if m dominates x in $G(V + W)$ **then**
          $c++$;
      if $c > c_{best}$ **then**
        $c_{best} := c$; $m_{best} := m$; $s_{best} := s$;
    for all $o_{-i} \in O_{-i}$ **do**
      $V_i(m_{best}, o_{-i})$+=max$(0, U_i(s_{best}, o_{-i}) - (U_i(m_{best}, o_{-i}) + V_i(m_{best}, o_{-i})))$;
return V;

$$G = \begin{array}{|c|c|c|c|} \hline 20 \; 0 & 11 \; 9 & 15 \; 15 & 15 \; 15 \\ \hline 11 \; 9 & 20 \; 0 & 15 \; 15 & 15 \; 15 \\ \hline 19 \; 10 & 10 \; 19 & 9 \; 11 & 0 \; 20 \\ \hline 10 \; 19 & 19 \; 10 & 0 \; 20 & 9 \; 11 \\ \hline \end{array} \qquad V = \begin{array}{|c|c|c|c|} \hline 0 \; \infty & 0 \; \infty & 0 \; 0 & 0 \; 0 \\ \hline 0 \; \infty & 0 \; \infty & 0 \; 0 & 0 \; 0 \\ \hline 1 \; 1 & 1 \; 1 & \infty \; 0 & \infty \; 0 \\ \hline 1 \; 1 & 1 \; 1 & \infty \; 0 & \infty \; 0 \\ \hline \end{array}$$

Fig. 3. 2-player game where O 's optimal implementation V yields a region $|X^*(V)| > 1$.

run in polynomial time. In order to study the quality of the resulting implementations, we performed several simulations comparing the implementation costs computed by the different algorithms. We have focused on two-person games using random game tables where both players have payoffs chosen uniformly at random from the interval $[0, max]$, for some constant max. We have also studied generalized *scissors, rock, paper games* (a.k.a., *Jan Ken Pon games*), that is, *symmetric zero-sum games* with payoff values chosen uniformly at random from an interval $[0, max]$. We find that – for the same interval and the same number of strategies – the average implementation cost of random symmetric zero-sum games, random symmetric games, and completely random games hardly deviate. This is probably due to the fact that in all examined types of random games virtually all strategies are non-dominated. Therefore, in the following, we present our results on symmetric random games only.

Algorithm 4 Reduction Algorithm $\mathcal{ALG}_{red}$

Input: Game G, convex target region O
Output: Implementation V of O

$[k, V] := greedy(G, O)$;
$k_{temp} := -1$; $c_i := \bot \; \forall i$; $T_i := \{\} \; \forall i$;
while $(k > 0) \wedge (\exists i : |O_i| > 1) \wedge (\exists i : O_i \not\subseteq T_i)$ **do**
  for all $i \in N$ **do**
    $x_i := \arg\min_{o_i \in O_i} (\max_{o_{-i} \in O_{-i}} U_i(o_i, o_{-i}))$;
    if $(O_i \not\subseteq T_i) \wedge (\neg c_j \forall j) \wedge (x_i \in T_i)$ **then**
      x_i:=$\arg\min_{o_i \in O_i \setminus \{x_i\}} (\max_{o_{-i} \in O_{-i}} (U_i(o_i, o_{-i})))$;
    if $|O_i| > 1$ **then**
      $O_i := O_i \setminus \{x_i\}$;
    $[k_{temp}, V] := greedy(G, O)$;
    if $k_{temp} \geq k$ **then**
      $O_i := O_i \cup \{x_i\}$; $T_i := T_i \cup \{x_i\}$; $c_i := \bot$;
    else
      $k := k_{temp}$; $T_i := \{\} \; \forall i$; $c_i := \top$;
return V;

Non-Exact Implementation We observe that implementing the best singleton often yields low costs. In other words, especially when large sets have to be implemented, our greedy algorithms tend to implement too many strategy profiles and consequently incur unnecessarily high costs. However, while this is often true in random games, there are counter examples where the cheapest singleton is costly compared to the implementation found by the greedy algorithms; Fig. 3 depicts a situation where the greedy algorithm computes a better solution. We presume that $\mathcal{ALG}_{red}$ might improve relatively to the best singleton heuristic algorithm for larger player sets.

Fig. 4 plots the implementation costs determined by the $\mathcal{ALG}_{greedy}$, $\mathcal{ALG}_{red}$ and the singleton algorithm as a function of the number of strategies involved. On average, the singleton algorithm performed much better than the other two, with $\mathcal{ALG}_{greedy}$ being the worst of the candidates. In the second plot we can see the implementation costs the algorithms compute for different payoff value intervals $[0, max]$. As expected, the implementation cost increases for larger intervals.

Exact Implementation The observation that the greedy algorithm $\mathcal{ALG}_{greedy}$ implements rather large subregions of O suggests that it may achieve good results for exact implementations. We can modify an implementation V of O, which yields a subset of O, without changing any entry $V_i(o), o \in O$, such that the resulting V implements O *exactly*.

Theorem 6. *If $O_{-i} \subset X_{-i} \; \forall i \in N$, it holds that $k^*(O) \leq \max_{o \in O} V(o)$ for an implementation V of O.*

PROOF. If V is a non-exact implementation of O, there are some strategies O_i dominated by other strategies in O_i. A dominated strategy o_i can be made non-dominated by adding payments to the existing V_i for profiles of the form $(o_i, \bar{o}_{-i})$, where $\bar{o}_{-i} \in$

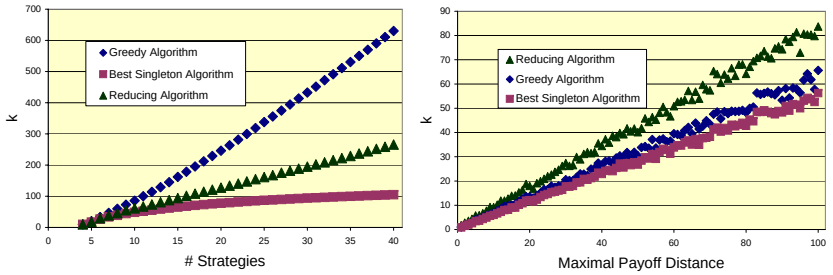

Fig. 4. The average implementation cost k of sets O over 100 random games where $|O_i| = \lfloor n/3 \rfloor$. *Left*: utility values chosen uniformly at random from $[0, 20]$. For different intervals we obtain approximately the same result when normalizing k with the maximal possible value. *Right*: eight strategies are used; other numbers of strategies yield similar results.

$X_{-i} \backslash O_{-i}$. Let $a \in O_i$ dominate $b \in O_i$ in $G(V)$. The interested party can annihilate this relation of a dominating b by choosing payment $V_i(b, \bar{o}_{-i})$ such that Player i's resulting payoff $U_i(b, \bar{o}_{-i}) + V_i(b, \bar{o}_{-i})$ is larger than $U_i(a, \bar{o}_{-i}) + V_i(a, \bar{o}_{-i})$ and therefore a does not dominate b anymore. As such, all dominations inside O_i can be neutralized even if $|X_{-i} \setminus O_{-i}| = 1$. One must realize that the relation of domination is *irreflexive* and *transitive* and therefore establishes a strict order among the strategies. Let $\bar{o}_{-i} \in X_{-i} \backslash O_{-i}$ be a column in Player i's payoff matrix outside O. By choosing the payments $V_i(o_i, \bar{o}_{-i})$ such that the resulting payoffs $U_i(o_i, \bar{o}_{-i}) + V_i(o_i, \bar{o}_{-i})$ establish the same order with the *less-than relation* ($<$) as the strategies o_i with the domination relation, all $o_i \in O_i$ will be non-dominated. Thus, a V' can be constructed from V which implements O exactly without modifying any entry $V_i(o), o \in O$. □

Theorem 6 enables us to use $\mathcal{ALG}_{greedy}$ for an exact cost approximation by simply computing $\max_{o \in O} V(o)$ instead of $\max_{x \in X^*(V)} V(x)$. Fig. 5 depicts the exact implementation costs determined by $\mathcal{ALG}_{greedy}$, $\mathcal{ALG}_{exact}$ and the perturbation algorithm from [11]. The first figure plots k as a function of the number of strategies, whereas the second figure demonstrates the effects of varying the size of the payoff interval. Due to the large runtime of $\mathcal{ALG}_{exact}$, we were only able to compute k for a small number of strategies. However, for these cases, our simulations reveals that $\mathcal{ALG}_{greedy}$ often finds implementations which are close to optimal and is better than the perturbation algorithm. For different payoff value intervals $[0, max]$, we observe a faster increase in k than in the non-exact implementation case. This suggests that implementing a smaller region entails lower costs for random games on average.

Furthermore, our simulations revealed that the variance of the cost found decreases with the number of strategies for all algorithms, while it remains roughly constant for intervals of various size. The variance of the singleton heuristic is typically smaller than the variance of $\mathcal{ALG}_{greedy}$ and $\mathcal{ALG}_{red}$. The same holds for exact implementations, where the perturbation algorithm has the largest variance.

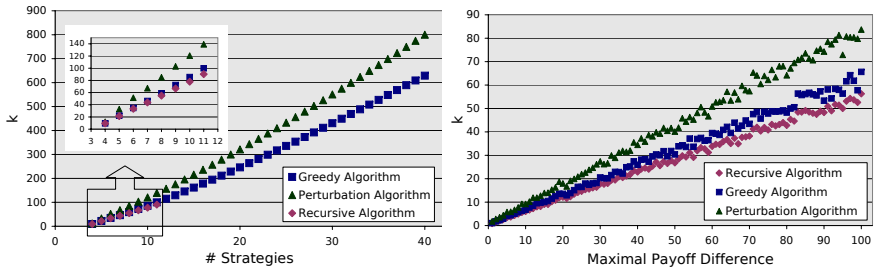

Fig. 5. The average exact implementation cost k of sets O over 100 random games where $|O_i| = \lfloor n/3 \rfloor$. *Left*: utility values chosen uniformly at random from $[0, 20]$. For other intervals we obtain approximately the same result when normalizing k with the maximal possible value. *Right*: eight strategies are used; the plot is similar for other numbers of strategies.

Finally, we tested different options to choose the next strategy in $\mathcal{ALG}_{greedy}$ (Line 8) and $\mathcal{ALG}_{red}$ (Lines 5 and 7). However, none of the alternatives we tested performed better than the ones described in Section 4.

In conclusion, our simulations have shown that for the case of non-exact implementations, there are interesting differences between the algorithms proposed in Section 4. In particular, the additional reductions by $\mathcal{ALG}_{red}$ are beneficial. For the case of exact implementations, our modified greedy algorithm yields good results. As a final remark we want to mention that, although $\mathcal{ALG}_{greedy}$ and $\mathcal{ALG}_{red}$ may find cheap implementations in the average case, there are examples where the approximation ratio of these algorithms is large.

6 Variations

Mechanism design by creditability offers many interesting extensions. In this section, two alternative models of rationality are introduced. If we assume that players do not just select *any* non-dominated strategy, but have other parameters influencing their decision process, our model has to be adjusted. In many (real world) games, players typically do not know which strategies the other players will choose. In this case, a player cannot do better than assume the other players to select a strategy *at random*. If a player wants to maximize her gain, she will take the *average payoff* of strategies into account. This kind of decision making is analyzed in the subsequent section. Afterwards, risk-averse players are examined. Finally, we take a brief look at the dynamics of repeated games with an interested third party offering payments *in each round*.

6.1 Average Payoff Model

As a player may choose any non-dominated strategy, it is reasonable to compute the payoff which each of her strategy will yield *on average*. Thus, assuming no knowledge on the payoffs of the other players, each strategy x_i has an average payoff of $p_i(x_i) := \frac{1}{|X_{-i}|} \sum_{x_{-i} \in X_{-i}} U_i(x_i, x_{-i})$ for Player i. Player i will then select the strategy $s \in X_i$

with the largest $p_i(s)$, i.e., $s = \arg\max_{s \in X_i} p_i(s)$. If multiple strategies have the same average payoff, she plays one of them uniformly at random. For such average strategy games, we say that x_i *dominates* x_i' iff $p_i(x_i) > p_i(x_i')$. Note that with this modified meaning of domination, the region of non-dominated strategies, X^*, differs as well.

The average payoff model has interesting properties, e.g., singleton profiles can be implemented for free.

Theorem 7. *If players maximize their average payoff, singleton strategy profiles are always 0-implementable if there are at least two players with at least two strategies.*

PROOF. Let z be the strategy profile to be implemented. In order to make Player i choose strategy z_i, the interested party may offer payments for any strategy profile $(z_i, \bar{z}_{-i})$ where $\bar{z}_{-i} \in X_{-i} \setminus \{z_{-i}\}$ such that $p_i(z_i)$ becomes Player i's largest average payoff in $G(V)$. Since each player has at least two strategies to choose from, there is at least one $\bar{z}_{-i}$, and by making $V_i(z_i, \bar{z}_{-i})$ large enough (e.g., $V_i(z_i, \bar{z}_{-i}) := \max_{x_i \in X_i} \sum_{x_{-i} \in X_{-i}} U_i(x_i, x_{-i}) + \epsilon$) this can always be achieved. Therefore, z can be implemented without promising any payments for z. □

Theorem 7 implies that entire strategy profile regions O are 0-implementable as well: we just have to implement any singleton inside O.

Corollary 3. *In average strategy games where every player has at least two strategies, every strategy profile region can be implemented for free.*

Exact implementations can be implemented at no costs as well.

Theorem 8. *In average strategy games where $O_{-i} \subset X_{-i} \ \ \forall i \in N$, each strategy profile region has an exact 0-implementation.*

PROOF. The mechanism designer can proceed as follows. Let $\mu_i := \max_{x_i \in X_i} \{p_i(x_i)\}$. We set $V_i(o_i, \bar{o}_{-i})$:=$|X_{-i}|(\mu_i - p_i(x_i)) - U_i(x_i, x_{-i}) + \epsilon, \forall o_i \in O_i, \bar{o}_{-i} \in X_{-i} \setminus O_{-i}$. Consequently, it holds that for each Player i and two strategies $x_i \in O_i$ and $x_i' \notin O_i$, $p_i(x_i) > p_i(x_i')$; moreover, no strategy $x_i \in O_i$ is dominated by any other strategy. As payments in $V_i(o_i, \bar{o}_{-i})$ with $o_i \in O_i$ and $\bar{o}_{-i} \in X_{-i} \setminus O_{-i}$ do not contribute to the implementation cost, Theorem 8 follows. □

6.2 Risk-Averse Players

Instead of striving for a high payoff on average, the players might be cautious or *risk-averse*. To account for such behavior, we adapt our model by assuming that the players seek to minimize the risk on missing out on benefits. In order to achieve this objective, they select strategies where the minimum gain is not less than any other strategy's minimum gain. If there is more than one strategy with this property, the risk-averse player can choose a strategy among these, where the average of the benefits is maximal. More formally, let $min_i := \max_{x_i \in X_i}(\min_{x_{-i} \in X_{-i}}(U_i(x_i, x_{-i})))$ and $\varnothing_X f(x) := \frac{1}{|X|} \cdot \sum_{x \in X} f(x)$. Then Player i selects a strategy m satisfying $m = \arg\max_{m \in M}(\varnothing_{X_{-i}} U_i(m, x_{-i}))$, where $M = \{x_i | \forall x_{-i} \ \ U_i(x_i, x_{-i}) = min_i\}$.

Theorem 9. *For risk-averse players the implementation cost of a singleton $z \in X$ is $k(z) = \sum_{i=1}^{n} \max(0, min_i - U_i(z))$*

PROOF. We show how to construct V implementing z with cost k and then prove that we cannot reduce the payments of $V(z)$. Since in this model every Player i makes her decision without taking into account the benefits other players might or might not obtain, it suffices to consider each V_i separately. To ensure that Player i selects z_i we have to set $V_i(z_i, x_{-i})$ to a value such that min_i is reached in $G(U+V)$ for each x_{-i}. Consequently we assign $V_i(z_i, x_{-i}) = \max(0, min_i - U_i(z_i, x_{-i}))$. We have to satisfy a second condition such that z_i is chosen, namely, $z_i = \arg\max_{m \in M}(\varnothing_{X_{-i}} U_i(m, x_{-i}))$. This is achieved by setting $V_i(z_i, x_{-i}) = \infty \ \ \forall x_{-i} \neq z_{-i}$. We repeat these steps for all Players i. Clearly, V constructed in this manner implements z. Since the cost k only comprises the additional payments in $V(z)$ and lowering $V_i(z)$ for any i results in Player i choosing a different strategy, we can deduce the statement of the theorem. □

For strategy profile regions, the situation with risk-averse players differs from the standard model considerably.

Theorem 10. *For risk-averse players the implementation cost for a strategy profile region $O \subset X$ is $k(O) = \min_{o \in O} \sum_{i=1}^{n} \max(0, min_i - U_i(o))$.*

PROOF. Since we have to add up the cost to reach the required minimum for every strategy profile in $X^*(V)$ it cannot cost less to exactly implement more than one strategy profile, i.e., find V such that $|X^*(V)| = 1$. Thus V implementing the "cheapest" singleton in O yields an optimal implementation for O, and the claim follows. □

Algorithm 5 $\mathcal{ALG}_{risk}$Risk-averse Players: Exact Implementation

Input: Game G, target region O, $O_i \cap X_i^* = \emptyset \ \forall i \in N$
Output: V

compute X^*;
$V_i(z) = 0$ for all $i \in N, z \in X$;
for all $i \in N$ **do**
  $V_i(x_i, x_{-i})$:=$\infty \ \ \forall x_i \in O_i, x_{-i} \in X_{-i} \setminus O_{-i}$;
  $V_i(x_i, x_{-i}) := \max(0, min_i - U_i(x_i, x_{-i})) \ \ \forall x_i \in O_i, x_{-i} \in X_{-i}$;
  if $O_{-i} = X_{-i}$ **then**
    if $\tau(O_i) > \tau(X_i^*)$ **then**
      if $|X_i| + \epsilon|O_i| > |X_i| + \sum_{o_i} \delta(o_i)$ **then**
        $V_i(o_i, x_{-i})$:=$V_i(o_i, x_{-i}) + \delta(o_i) \ \ \forall o_i, x_{-i}$;
      else
        $V_i(o_i, x_{-i})$:=$V_i(o_i, x_{-i}) + \epsilon \ \ \forall o_i, x_{-i}$;
    else
      if $\epsilon|O_i| > \sum_{o_i} [\epsilon + \delta(o_i)]$ **then**
        $V_i(o_i, x_{-i})$:=$V_i(o_i, x_{-i}) + \epsilon + \delta(o_i) \ \ \forall o_i, x_{-i}$;
      else
        $V_i(o_i, x_{-i})$:=$V_i(o_i, x_{-i}) + \epsilon \ \ \forall o_i, x_{-i}$;
return V;

In Section 4, we conjectured the problem of computing $k(O)$ to be **NP**-complete for both general and exact implementations. This is not the case for risk-averse players, as the following theorem states.

Theorem 11. *$\mathcal{ALG}_{risk}$ computes $k(O)$ in time $\mathrm{O}\big(n|X|^2\big)$, thus the problem of computing k for risk-averse agents is in* $\mathbf{P}$.

PROOF. For the non-exact case this theorem follows directly from Theorem 10. In order to prove Theorem 11 for exact implementations, we demonstrate how to compute $V_i(o)$ such that V implements the entire region O optimally. For a Player i and a set of strategies $Y_i \subseteq X_i$, we define $\tau(Y_i)$:=$\max_{x_i \in Y_i}(\varnothing_{X_{-i}}((U+V)_i(x_i, x_{-i})))$ to be the maximum of the average benefits over all strategies. For each strategy of a Player i, we define $\delta(x_i)$:=$\max(\tau(O_i), \tau(X_i^*)) - \varnothing_{X_{-i}}((U+V)_i(x_i, x_{-i})))$, for $x_i \in X_i$, to be the difference of the averages. Algorithm 5 constructs V if the target region O and X^* are disjoint. Analogously to the proofs above we can deal with each Player i individually. The algorithm computes for all cases how much the interested party has to offer at least for strategy profiles in O and sets $V_i(x_i, x_{-i})$ to infinity for all $x_i \in O_i$, $x_{-i} \in X_{-i} \setminus O_{-i}$ (Line 4). Then, for each Player i, strategies O_i have to reach the minimum payoff of strategies in X_i^*. This suffices for an exact implementation if $O_i \subset X_i$, i.e. if there exists at least one strategy $x_i \notin O_i$. Otherwise, we determine whether it costs more to exceed the minimum constraint or the average constraint for all O_i if O_i covers whole columns and adjust V_i accordingly. Thus the algorithm ensures that only strategies in O are chosen while all strategies in O are selected.

The algorithm can be extended easily to work for instances where $X_i^* \subset O_i$. As the extension is straight-forward and does not provide any new insights, we omit it. The runtime of the algorithm can be determined to be $\mathrm{O}\big(n|X|^2\big)$, thus we can compute $k^*(O) = \max_{o \in O} V(o)$ in polynomial time. □

6.3 Round-based Mechanisms

The previous sections dealt with static models only. Now, we extend our analysis to dynamic, round-based games, where the designer offers payments to the players after each round in order to make them change strategies. This opens many questions: For example, imagine a concrete game such as a *network creation game* [7] where all players are stuck in a costly Nash equilibrium. The goal of a mechanism designer could then be to guide the players into another, better Nash equilibrium. Many such extensions are reasonable; due to space constraints, only one model is presented in more detail.

In a dynamic game, we regard a strategy profile as a state in which the participants find themselves. In a network context, each $x \in X$ could represent one particular network topology. We presume to find the game in an initial starting state $s^{T=0} \in X$ and that, in state $s^{T=t}$, each Player i only sees the states she can reach by changing her strategy given the other players remain with their chosen strategies. Thus Player i sees only strategy profiles in $X_{visible,i}^{T=t} = X_i \times \{s_{-i}^{T=t}\}$ in round t. In every round t, the mechanism designer offers the players a payment matrix $V^{T=t}$ (in addition to the game's static payoff matrix U). Then all players switch to their best visible strategy (which is any best response $B_i(s_{-i}^{T=t})$), and the game's state changes to $s^{T=t+1}$. Before the next round starts, the mechanism designer disburses the payments $V^{T=t}(s^{T=t+1})$ offered for the newly reached state. The same procedure is repeated until the mecha-

nism designer decides to stop the game. We prove that a mechanism designer can guide the players to any strategy profile at zero costs in two rounds.

Theorem 12. *Starting in an arbitrary strategy profile, a dynamic mechanism can be designed to lead the players to any strategy profile without any expenses in at most two rounds if* $|X_i| \geq 3 \; \forall i \in N$.

In order to simplify the proof we begin with a helper lemma. Let $X_{visible}^{T=t}$ denote the *visible strategy profile region* in round t, i.e., $X_{visible}^{T=t} = \bigcup_{i=1}^{n} X_{visible,i}^{T=t}$.

Lemma 1. *The third party can lead the players of a dynamic game to any strategy profile outside the visible strategy profile region without any expenses in one round.*

PROOF. Let $s \in X$ be the starting strategy profile and e the desired end strategy profile in the non-visible region of s. The designer can implement e in just one round by offering each Player i an infinite amount $V_i(x)$ for the strategy profile $x = (e_i, s_{-i})$ and zero for any other. Thus each player will switch to e_i. Since $V_i(e_i, s_{-i})$ are the only positive payments offered and since all $x = (e_i, s_{-i})$ are visible and e is non-visible from s, e is implemented at no cost. □

PROOF OF THEOREM 12. Consider an arbitrary starting strategy profile s and a desired strategy profile e. If e is not visible from s, e is implementable at no cost in one round, as seen in Lemma 1. If e is visible from s, the interested party can still implement e for free by taking a detour to a strategy profile d which is neither in s' visible region nor in e's visible region. Such a strategy profile d exists if Player i who sees e from s has at least 3 strategies to choose from and $|X_{-i}| \geq 2$. □

See Fig. 6 for an illustration.

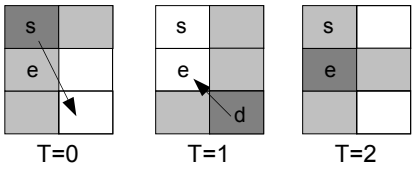

Fig. 6. A dynamic game: Starting in **s**, strategy profile **e** can be implemented at zero cost within two rounds by taking a detour to **d**. The colored region marks the visible strategy profiles at each step.

7 Conclusion

Rendering distributed systems robust to non-cooperative behavior has become an important research topic. This paper has studied the fundamental question: Which outcomes can be implemented by promising players money while the eventual payments are bounded? We have presented algorithms for various objectives yielding implementations of low cost. Furthermore, it has been shown that a greedy algorithm performs well for random games. Finally, this paper has initiated the study of risk-averse players and round-based games.

References

1. J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Partition Problem. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 43–52, 2005.
2. M. Babaioff, M. Feldman, and N. Nisan. Combinatorial Agency. In *Proc. 7th ACM Conference on Electronic Commerce (EC)*, pages 18–28, 2006.
3. G. Christodoulou and E. Koutsoupias. The Price of Anarchy of Finite Congestion Games. In *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–73, 2005.
4. R. Cole, Y. Dodis, and T. Roughgarden. How Much Can Taxes Help Selfish Routing? In *Proc. 4th ACM Conference on Electronic Commerce (EC)*, pages 98–107, 2003.
5. R. Cole, Y. Dodis, and T. Roughgarden. Pricing Network Edges for Heterogeneous Selfish Users. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 521–530, 2003.
6. R. Dash, D. Parkes, and N. Jennings. Computational Mechanism Design: A Call to Arms. In *IEEE Intelligent Systems*, 2003.
7. A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proc. 22nd Annual Symposium on Principles of Distributed Computing (PODC)*, pages 347–351, 2003.
8. D. S. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
9. L. Lovász. On the Ratio of Optimal Integral and Fractional Covers. *Discrete Mathematics*, 13:391–398, 1975.
10. E. Maskin and T. Sjöström. *Handbook of Social Choice Theory and Welfare (Implementation Theory)*, volume 1. North-Holland, Amsterdam, 2002.
11. D. Monderer and M. Tennenholtz. k-Implementation. In *Proc. 4th ACM Conference on Electronic Commerce (EC)*, pages 19–28, 2003.
12. T. Roughgarden. Stackelberg Scheduling Strategies. In *Proc. 33rd ACM Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.

Appendix

Non-Optimality of the Perturbation Algorithm

In this section, we give an example demonstrating that the optimal perturbation algorithm presented in [11] is not correct. The algorithm computes the payoff matrix V for the following game G with X^* and O and payoff matrices $V1$, $V2$.

G

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

V_1

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

V_2

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

As can be verified easily, V_1 implements O with cost $k = 3$. The payoff matrix V_2 computed by the optimal perturbation algorithm implements O as well, however, it has cost $k = 5$. The set of possible differences between an agent's payoffs in the original game for G is $E = \{0, 2, 3, 4\}$. We execute Lines 2 and 3 and obtain a matrix of perturbation G'. We have to go through the Lines 4 to 8 for Player p_1 twice, for

$e_1 = 0$ and $e_2 = 2$, generating $G'(p_1, e_1), G'(p_1, e_2)$ respectively, such that afterwards $(G + G')^*_1$ coincides with O_1. Executing Lines 9 to 13 for Player p_2 until the condition in Line 13 is satisfied and hence $(G + G')^*_2 \equiv O_2$ results in $G'(p_2, e_1) = G'(p_1, e_2)$. Thus, the perturbation algorithm returns the payoff matrix $V = G'$.

G'

0 0	5 0
0 0	5 0
0 5	0 0

$G'(p_1, e_1)$

0 0	5 0
0 0	5 0
0 5	0 0

$G'(p_1, e_2)$

2 0	5 0
2 0	5 0
0 5	0 0

$G'(p_2, e_2)$

2 2	5 0
0 2	5 0
0 5	0 0

$G'(p_2, e_3)$

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