A Combinatorial Algorithm for Strong Implementation of Social Choice Functions

Clemens Thielen and Stephan Westphal

Abstract

We consider algorithmic aspects of the classical mechanism design problem of implementing social choice functions. We show how an adaption of the well-known negative cycle criterion for weak implementability can be used to decide the question of implementability in the strong sense when one restricts to incentive compatible direct revelation mechanisms. We derive an efficient combinatorial algorithm that computes the payments of an incentive compatible direct revelation mechanism that strongly implements a given social choice function in dominant strategies or decides that none exist.

Our result complements the results obtained in the companion paper of Krumke and Thielen [3], where a nondeterministic polynomial time algorithm for the more general problem of deciding of strong implementability via *indirect* mechanisms is given. This more general problem is expected to be NP-complete.

1 Introduction

One of the central problems considered in mechanism design is the implementation of social choice functions. In this problem, there are n selfish agents, which must make a collective decision from some finite set X of possible social choices (or *outcomes*). Each agent i has a private value θ_i (called the agent's *type*) that belongs to a finite set Θ_i (called the agent's *type space*) and influences the preferences of all agents over the alternatives in X. Formally, this is modelled by a *valuation function* $V_i : X \times \Theta \to \mathbb{Q}$ for each agent i, which specifies a valuation $V_i(x, \theta)$ that agent i assigns to outcome $x \in X$ when the vector of types of all agents is $\theta \in \Theta = \Theta_1 \times \cdots \times \Theta_n$. The type space Θ_i of agent i is public knowledge, but only agent i knows the true value of θ_i. Every agent i reports a claimed value $\theta'_i \in \Theta_i$ (a *bid*) for her type, and the resulting collective decision is given by a social choice function $f : \Theta \to X$ that maps vectors of bids of the agents to outcomes in X. A *mechanism* $\Gamma_{(f,P)}$ in this setting is given by a payment $P_i(\theta')$ to each agent i that depends on the vector θ' of bids and is used to motivate the agents to report their types truthfully.

When the concept of *weak* implementation is used, a mechanism is said to *implement* the social choice function f if truthfully reporting her type is a *dominant strategy* for every agent, i.e., it maximizes the sum of the agent's valuation and her payment for every possible behavior of the other agents and for every possible vector θ of true types.[†] The more robust concept of implementation called *strong implementation* (also known as *full implementation*) additionally requires that all other dominant strategy equilibria of the mechanism yield the same outcomes as truthful reporting, so the desired social choices are obtained independently of the equilibrium that is actually played by the agents.

It is easy to see that weak implementation of a given social choice function can be expressed as a system of linear inequalities, in which the variables correspond to the payments. Rochet [6] observed that this system can be interpreted as the problem of finding node potentials in complete, directed graphs on the agents' type spaces with changes of valuations as arc weights. Hence, an implementation exists if and only if there is no negative cycle in

[†]Note that there are also other notions of implementation, e.g., implementation in Bayes Nash equilibrium. In this paper, we only consider implementation in dominant strategies.

these graphs. Later, it was shown by Gui et al.[2] that it suffices to consider node potentials in smaller graphs on the set X of possible outcomes.

In this paper, we show how the above node potential interpretations of the weak implementability problem can be adapted for deciding also *strong* implementation. Here, some of the inequalities in the linear system have to be fulfilled with strict inequality, i.e., a point in the relative interior of the corresponding polyhedron is sought. We show how such a point can be found by an efficient combinatorial algorithm that perturbates a node potential corresponding to a weak implementation such that the reduced cost of some arcs in the graphs becomes strictly positive, which corresponds to the strict inequalities in the system. To do so, all arcs whose inequalities are already strictly fulfilled are deleted and depth first search is used to find nodes with no outgoing arcs, whose potential can then be perturbated. Furthermore, we use contraction techniques to handle cycles of weight zero. Using these methods, our algorithm computes the payments of a strong implementation of the given social choice function or decides that none exist. The running time is linear in $|\Theta|$, which usually is the largest part of the input. In public project settings, for example, $|\Theta|$ can be quite large, whereas $|X|$ is usually two (the project is either done or not).

We remark that there is also a more general definition of a mechanism, where each agent i is allowed to bid a value s_i from an arbitrary set S_i of bids instead of just reporting a claimed value for her type. A classical result known as the *Revelation Principle* (cf. [4, p. 871]) states that, when considering *weak* implementation, it imposes no loss of generality to restrict to *incentive compatible direct revelation mechanisms* as defined above. For *strong* implementation, it is known that it suffices to consider *augmented revelation mechanisms*, in which the set Θ_i of types of each agent i is a subset of the set S_i of her possible bids (cf. [5] for Bayesian equilibria and the companion paper of Krumke and Thielen [3] for dominant strategies). Most strongly implementable social choice functions can, however, be strongly implemented via incentive compatible direct revelation mechanisms. Thus, since the general problem of deciding strong implementability via augmented revelation mechanisms is expected to be computationally intractable (until recently, it was not even known to belong to NP and it is suspected to be NP-complete, cf. [3]), it makes sense to restrict to incentive compatible direct revelation mechanisms also for strong implementation.

2 The Algorithm

We now present our algorithm for strong implementation of social choice functions. Formally, the problem is defined as follows:

Definition 1 (The Strong Implementability Problem).
INSTANCE: *The number n of agents, the set X of possible social choices, the sets Θ_i of possible types of the agents, the valuation functions $V_i : X \times \Theta \to \mathbb{Q}$, and the social choice function $f : \Theta \to X$.*
TASK: *Compute payments $P_i : \Theta \to \mathbb{Q}$, $i = 1, \dots, n$, to the agents such that the mechanism $\Gamma_{(f,P)}$ strongly implements f, or decide that none exist.*

The encoding length of an instance of Strong Implementability can be calculated as follows: For every valuation function $V_i : X \times \Theta \to \mathbb{Q}$, we need to store $|X| \cdot |\Theta|$ rational numbers. The social choice function $f : \Theta \to X$ has encoding length $|\Theta| \cdot \log(|X|)$. Thus, the encoding length of an instance of Strong Implementability is in $\Omega(|X| \cdot |\Theta| \cdot n)$.

We start our analysis by formulating a system of linear inequalities whose solutions correspond to the values of payment functions P_i needed to implement a social choice function f. Denoting the $(n-1)$-dimensional vector resulting from an n-vector v when the i-th component is deleted by $v_{-i} := (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$, this linear system in the variables $P_i(\theta')$, $i = 1, \dots, n$, $\theta' \in \Theta$, can be written as follows:

For all $i \in N$, $(\theta_i, \theta'_i) \in \Theta_i^2$, and $\theta_{-i}, \theta'_{-i} \in \Theta_{-i}$:

$$V_i(f(\theta'_i, \theta'_{-i}), \theta) + P_i(\theta'_i, \theta'_{-i}) \leq V_i(f(\theta_i, \theta'_{-i}), \theta) + P_i(\theta_i, \theta'_{-i}) \tag{1}$$

For all $i \in N$ and $(\theta_i, \theta'_i) \in \Theta_i^2$ with $f(\theta_i, \bar{\theta}_{-i}) \neq f(\theta'_i, \bar{\theta}_{-i})$ for *some* $\bar{\theta}_{-i} \in \Theta_{-i}$, there exists $\theta_{-i}, \theta'_{-i} \in \Theta_{-i}$ such that:

$$V_i(f(\theta'_i, \theta'_{-i}), \theta) + P_i(\theta'_i, \theta'_{-i}) < V_i(f(\theta_i, \theta'_{-i}), \theta) + P_i(\theta_i, \theta'_{-i}) \tag{2}$$

Here, the Inequalities (1) encode that truthfully reporting her type is a dominant strategy for each agent i: For every type θ_i of agent i, reporting θ_i truthfully is at least as good as reporting any other possible type θ'_i, no matter what the type vector θ_{-i} and the bid vector θ'_{-i} of the other agents are. Hence, the first half of the system encodes that the social choice function f is *weakly* implemented by the mechanism $\Gamma_{(f,P)}$. For *strong* implementation, the payments must additionally satisfy the strict Inequalities (2), which encode that there are no dominant strategies for any agent that yield outcomes different from the ones obtained by truthful reporting: If the second condition in the system was violated for some pair (θ_i, θ'_i), bidding θ'_i would always be optimal for agent i when her type is θ_i, and she could change the outcome by bidding θ'_i instead of θ_i in the case that the vector of types of the other agents is $\bar{\theta}_{-i}$ and they report their types truthfully.

We now reformulate the system in order to be able to solve it efficiently via shortest path computations in directed graphs. For every agent $i \in N$ and every fixed pair $(\theta_{-i}, \theta'_{-i}) \in \Theta_{-i}^2$ of a type vector and a bid vector of the other agents, we define a function $c_i^{(\theta_{-i}, \theta'_{-i})} : \Theta_i^2 \to \mathbb{Q}$ by

$$c_i^{(\theta_{-i}, \theta'_{-i})}(\theta_i, \theta'_i) := V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(f(\theta'_i, \theta'_{-i}), \theta) \quad \forall\, \theta_i, \theta'_i \in \Theta_i.$$

Using this notation, we can rewrite the above system of inequalities as follows:

For all $i \in N$, $(\theta_i, \theta'_i) \in \Theta_i^2$, and $\theta_{-i}, \theta'_{-i} \in \Theta_{-i}$:

$$P_i(\theta'_i, \theta'_{-i}) - P_i(\theta_i, \theta'_{-i}) \leq c_i^{(\theta_{-i}, \theta'_{-i})}(\theta_i, \theta'_i) \tag{3}$$

For all $i \in N$ and $(\theta_i, \theta'_i) \in \Theta_i^2$ with $f(\theta_i, \bar{\theta}_{-i}) \neq f(\theta'_i, \bar{\theta}_{-i})$ for *some* $\bar{\theta}_{-i} \in \Theta_{-i}$, there exists $\theta_{-i}, \theta'_{-i} \in \Theta_{-i}$ such that:

$$P_i(\theta'_i, \theta'_{-i}) - P_i(\theta_i, \theta'_{-i}) < c_i^{(\theta_{-i}, \theta'_{-i})}(\theta_i, \theta'_i) \tag{4}$$

Observe that the left-hand sides of the Inequalities (3) and (4) are independent of the type vector θ_{-i} of all agents except i. Defining

$$\underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) := \min_{\theta_{-i} \in \Theta_{-i}} c_i^{(\theta_{-i}, \theta'_{-i})}(\theta_i, \theta'_i) = \min_{\theta_{-i} \in \Theta_{-i}} \Big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(f(\theta'_i, \theta'_{-i}), \theta) \Big)$$

and

$$\bar{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) := \max_{\theta_{-i} \in \Theta_{-i}} c_i^{(\theta_{-i}, \theta'_{-i})}(\theta_i, \theta'_i) = \max_{\theta_{-i} \in \Theta_{-i}} \Big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(f(\theta'_i, \theta'_{-i}), \theta) \Big)$$

for $\theta_i, \theta'_i \in \Theta_i$, we can, thus, rewrite the system of inequalities as

For all $i \in N$, $(\theta_i, \theta'_i) \in \Theta_i^2$, and $\theta'_{-i} \in \Theta_{-i}$:

$$P_i(\theta'_i, \theta'_{-i}) - P_i(\theta_i, \theta'_{-i}) \leq \underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) \tag{5}$$

For all $i \in N$ and $(\theta_i, \theta'_i) \in \Theta_i^2$ with $f(\theta_i, \bar{\theta}_{-i}) \neq f(\theta'_i, \bar{\theta}_{-i})$ for *some* $\bar{\theta}_{-i} \in \Theta_{-i}$, there exists $\theta'_{-i} \in \Theta_{-i}$ such that:

$$P_i(\theta'_i, \theta'_{-i}) - P_i(\theta_i, \theta'_{-i}) < \bar{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) \tag{6}$$

Observe that, whenever $\underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i) < \bar{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i)$ for some $\theta'_{-i} \in \Theta_{-i}$, the second condition follows automatically from the first one for this pair (θ_i,θ'_i). Hence, the system reduces to

For all $i \in N$, $(\theta_i,\theta'_i) \in \Theta_i^2$, and $\theta'_{-i} \in \Theta_{-i}$:

$$P_i(\theta'_i,\theta'_{-i}) - P_i(\theta_i,\theta'_{-i}) \leq \underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i) \tag{7}$$

For all $i \in N$ and $(\theta_i,\theta'_i) \in \Theta_i^2$ with $f(\theta_i,\bar{\theta}_{-i}) \neq f(\theta'_i,\bar{\theta}_{-i})$ for *some* $\bar{\theta}_{-i} \in \Theta_{-i}$ and $\underline{c}_i^{\bar{\theta}_{-i}}(\theta_i,\theta'_i) = \bar{c}_i^{\bar{\theta}_{-i}}(\theta_i,\theta'_i)$ *for all* $\bar{\theta}_{-i} \in \Theta_{-i}$ with $f(\theta_i,\bar{\theta}_{-i}) \neq f(\theta'_i,\bar{\theta}_{-i})$, there exists $\theta'_{-i} \in \Theta_{-i}$ such that $f(\theta_i,\theta'_{-i}) \neq f(\theta'_i,\theta'_{-i})$ and

$$P_i(\theta'_i,\theta'_{-i}) - P_i(\theta_i,\theta'_{-i}) < \underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i). \tag{8}$$

Moreover, for every fixed agent i and $\theta'_{-i} \in \Theta_{-i}$, consider a pair $(\theta_i,\theta'_i) \in \Theta_i^2$ of types of agent i such that $f(\theta_i,\theta'_{-i}) = f(\theta'_i,\theta'_{-i}) =: x \in X$. Then we have

$$\begin{aligned}\underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i) &= \min_{\theta_{-i}\in\Theta_{-i}} \Big(V_i(f(\theta_i,\theta'_{-i}),\theta) - V_i(f(\theta'_i,\theta'_{-i}),\theta)\Big) \\ &= \min_{\theta_{-i}\in\Theta_{-i}} \Big(V_i(x,\theta) - V_i(x,\theta)\Big) = 0\end{aligned}$$

and analogously $\underline{c}_i^{\theta'_{-i}}(\theta'_i,\theta_i) = 0$. Hence, the Inequalities (7) corresponding to (θ_i,θ'_i) imply that

$$P_i(\theta'_i,\theta'_{-i}) - P_i(\theta_i,\theta'_{-i}) \quad \leq \quad 0 \quad \text{and} \quad P_i(\theta_i,\theta'_{-i}) - P_i(\theta'_i,\theta'_{-i}) \leq 0,$$

which yields $P_i(\theta'_i,\theta'_{-i}) = P_i(\theta_i,\theta'_{-i})$. Thus, for fixed i and fixed $\theta'_{-i} \in \Theta_{-i}$, the payment $P_i(\theta_i,\theta'_{-i})$ does in fact only depend on the outcome $f(\theta_i,\theta'_{-i})$ chosen when agent i bids θ_i. Hence, for every $x \in X$ that results as the outcome $f(\theta_i,\theta'_{-i})$ for some $\theta_i \in \Theta_i$, we can define

$$P_i^{\theta'_{-i}}(x) := P_i(\theta_i,\theta'_{-i}) \quad \text{for some } \theta_i \in \Theta_i \text{ with } f(\theta_i,\theta'_{-i}) = x.$$

Writing

$$C(\theta_i,\theta'_i) := \{\theta'_{-i} \in \Theta_{-i} : f(\theta_i,\theta'_{-i}) \neq f(\theta'_i,\theta'_{-i})\},$$

we can, thus, write our system of inequalities as

For all $i \in N$, $(\theta_i,\theta'_i) \in \Theta_i^2$, and $\theta'_{-i} \in \Theta_{-i}$ with $f(\theta_i,\theta'_{-i}) \neq f(\theta'_i,\theta'_{-i})$:

$$P_i^{\theta'_{-i}}(f(\theta'_i,\theta'_{-i})) - P_i^{\theta'_{-i}}(f(\theta_i,\theta'_{-i})) \leq \underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i) \tag{9}$$

For all $i \in N$ and $(\theta_i,\theta'_i) \in \Theta_i^2$ with $C(\theta_i,\theta'_i) \neq \emptyset$ and $\underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i) = \bar{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i)$ for all $\theta'_{-i} \in C(\theta_i,\theta'_i)$: There exists $\theta'_{-i} \in C(\theta_i,\theta'_i)$ such that

$$P_i^{\theta'_{-i}}(f(\theta'_i,\theta'_{-i})) - P_i^{\theta'_{-i}}(f(\theta_i,\theta'_{-i})) < \underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i). \tag{10}$$

As observed above, the left-hand sides of Inequalities (9) and (10) are now independent of θ_i and θ'_i as long as the respective outcomes $f(\theta_i,\theta'_{-i})$ and $f(\theta'_i,\theta'_{-i})$ do not change. Defining $K^{\theta'_{-i}}(x) := \{\theta_i \in \Theta_i : f(\theta_i,\theta'_{-i}) = x\}$ for every $x \in X$, $W(\theta'_{-i}) := \{x \in X : K^{\theta'_{-i}}(x) \neq \emptyset\}$, and

$$c_i^{\theta'_{-i}}(x,x') := \min_{\substack{\theta_i,\theta'_i\in\Theta_i: \\ f(\theta_i,\theta'_{-i})=x \\ f(\theta'_i,\theta'_{-i})=x'}} \underline{c}_i^{\theta'_{-i}}(\theta_i,\theta'_i)$$

for all $x, x' \in W(\theta'_{-i})$, the system can be written as

For all $i \in N$, $\theta'_{-i} \in \Theta_{-i}$, and $x, x' \in W(\theta'_{-i})$ with $x \neq x'$:

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) \leq c_i^{\theta'_{-i}}(x, x') \tag{11}$$

For all $i \in N$ and $(\theta_i, \theta'_i) \in \Theta_i^2$ with $C(\theta_i, \theta'_i) \neq \emptyset$ and $\underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) = \bar{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i)$ for all $\theta'_{-i} \in C(\theta_i, \theta'_i)$: There exists $\theta'_{-i} \in C(\theta_i, \theta'_i)$ such that $\theta_i \in K^{\theta'_{-i}}(x), \theta'_i \in K^{\theta'_{-i}}(x')$ and

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) < c_i^{\theta'_{-i}}(\theta_i, \theta'_i). \tag{12}$$

Now observe that, whenever $\theta_i \in K^{\theta'_{-i}}(x), \theta'_i \in K^{\theta'_{-i}}(x')$, and $c_i^{\theta'_{-i}}(x, x') < \underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i)$ for *some* $\theta'_{-i} \in \Theta_{-i}$, the second condition follows automatically from the first one for this pair (θ_i, θ'_i). Hence, we just have to consider the second condition for the pairs (θ_i, θ'_i) for which $c_i^{\theta'_{-i}}(x, x') = \underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i)$ for *all* $\theta'_{-i} \in \Theta_{-i}$, x, x' with $\theta_i \in K^{\theta'_{-i}}(x)$ and $\theta'_i \in K^{\theta'_{-i}}(x')$. Thus, the system can be rewritten as

For all $i \in N$, $\theta'_{-i} \in \Theta_{-i}$, and $x, x' \in W(\theta'_{-i})$ with $x \neq x'$:

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) \leq c_i^{\theta'_{-i}}(x, x') \tag{13}$$

For all $i \in N$ and $(\theta_i, \theta'_i) \in \Theta_i^2$ with $C(\theta_i, \theta'_i) \neq \emptyset$, $\underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) = \bar{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i)$ for all $\theta'_{-i} \in C(\theta_i, \theta'_i)$, and $c_i^{\theta'_{-i}}(x, x') = \underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i)$ for all $\theta'_{-i} \in C(\theta_i, \theta'_i)$, $x, x' \in W$ with $\theta_i \in K^{\theta'_{-i}}(x)$ and $\theta'_i \in K^{\theta'_{-i}}(x')$: There exists $\theta'_{-i} \in C(\theta_i, \theta'_i)$ such that $\theta_i \in K^{\theta'_{-i}}(x), \theta'_i \in K^{\theta'_{-i}}(x')$ and

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) < c_i^{\theta'_{-i}}(x, x'). \tag{14}$$

Finally, the second condition in the system can be reformulated as follows: All conditions on the pairs (θ_i, θ'_i) do in fact only depend on the second value θ'_i through the corresponding outcomes $f(\theta'_i, \theta'_{-i})$. In particular, the values

$$\begin{aligned} \underline{c}_i^{\theta'_{-i}}(\theta_i, x') &:= \underline{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) \text{ for some } \theta'_i \in K^{\theta'_{-i}}(x') \\ \bar{c}_i^{\theta'_{-i}}(\theta_i, x') &:= \bar{c}_i^{\theta'_{-i}}(\theta_i, \theta'_i) \text{ for some } \theta'_i \in K^{\theta'_{-i}}(x') \end{aligned}$$

are well-defined for all $x \in W(\theta'_{-i})$. Defining

$$C(\theta_i, x') := \{\theta'_{-i} \in \Theta_{-i} : x' \in W(\theta'_{-i}), f(\theta_i, \theta'_{-i}) \neq x'\},$$

we can, hence, rewrite the system as

For all $i \in N$, $\theta'_{-i} \in \Theta_{-i}$, and $x, x' \in W(\theta'_{-i})$ with $x \neq x'$:

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) \leq c_i^{\theta'_{-i}}(x, x') \tag{15}$$

For all $i \in N$, $\theta_i \in \Theta_i$, and $x' \in X$ such that $C(\theta_i, x') \neq \emptyset$, $\underline{c}_i^{\theta'_{-i}}(\theta_i, x') = \bar{c}_i^{\theta'_{-i}}(\theta_i, x')$ for all $\theta'_{-i} \in C(\theta_i, x')$, and $c_i^{\theta'_{-i}}(f(\theta_i, \theta'_{-i}), x') = \underline{c}_i^{\theta'_{-i}}(\theta_i, x')$ for all $\theta'_{-i} \in C(\theta_i, x')$: There exists $\theta'_{-i} \in C(\theta_i, x')$ such that $f(\theta_i, \theta'_{-i}) = x$ and

$$P_i^{\theta'_{-i}}(x') - P_i^{\theta'_{-i}}(x) < c_i^{\theta'_{-i}}(x, x'). \tag{16}$$

Having reformulated the system as above, we can now solve it efficiently via shortest path computations. For every agent i and every fixed vector $\theta'_{-i} \in \Theta_{-i}$ of bids of the other agents, the Inequalities (15) corresponding to i and θ'_{-i} are exactly equivalent to the values $P_i^{\theta'_{-i}}(x)$, $x \in W(\theta'_{-i})$, defining a node potential in the complete, directed graph $G_i(\theta'_{-i})$ on the set $W(\theta'_{-i})$ with the cost of the arc from outcome x to x' given as $c_i^{\theta'_{-i}}(x, x')$. Thus, we can compute a solution $P_i^{\theta'_{-i}}(x)$, $x \in W(\theta'_{-i})$, of the Inequalities (15) by computing the shortest path distances from an arbitrary node x to all other nodes in the graph $G_i(\theta'_{-i})$ for every $i \in N$ and every $\theta'_{-i} \in \Theta_{-i}$. This can be done efficiently with the Bellman-Ford Algorithm (cf. for example [7]). In the case that one of the graphs $G_i(\theta'_{-i})$ contains a negative cycle (so we cannot compute a node potential in this graph), the Inequalities (15) do not have a solution, which implies that the given social choice function f cannot be implemented at all (not even *weakly*). Otherwise, the shortest path distances $P_i^{\theta'_{-i}}(x)$, $x \in W(\theta'_{-i})$, computed by the Bellman-Ford Algorithm yield payments $P_i(\theta')$ solving Inequalities (1) of the original system (and, thus, weakly implementing the social choice function f) by setting $P_i(\theta') := P_i^{\theta'_{-i}}(f(\theta'))$ for all $i = 1, \ldots, n$, $\theta' \in \Theta$.

For strong implementation, we now show how we can modify a solution of the Inequalities (15) obtained by shortest path computations such that it also satisfies the strict Inequalities (16) and, hence, corresponds to payments P of a mechanism $\Gamma_{(f,P)}$ that strongly implements f. This is done via the following procedure:

For every agent i and every $\theta'_{-i} \in \Theta_{-i}$, we again consider the graph $G_i(\theta'_{-i})$ and the corresponding payments $P_i^{\theta'_{-i}}(x)$ for $x \in W(\theta'_{-i})$. We delete all arcs (x, x') from $G_i(\theta'_{-i})$ for which the corresponding Inequality (15) is already fulfilled with strict inequality. After doing so, we also delete all isolated nodes, i.e., all nodes x with empty adjacency list $\mathrm{Adj}(x)$. In the remaining graph, which contains only arcs for which the corresponding Inequality (15) holds with equality, we then search for a node x with no outgoing arcs, i.e., with $\mathrm{Adj}^+(x) = \emptyset$. For such a node x, the value

$$\epsilon(x) := \min_{x' \in W(\theta'_{-i})} \left(c_i^{\theta'_{-i}}(x, x') - P_i^{\theta'_{-i}}(x') + P_i^{\theta'_{-i}}(x) \right)$$

is strictly positive, so we can lower $P_i^{\theta'_{-i}}(x)$ by $\epsilon(x)/2$ without violating any of the inequalities in our system. After doing so, all inequalities which were fulfilled with strict inequality before are still fulfilled with strict inequality, but, additionally, all the inequalities corresponding to arcs with end node x are now fulfilled with strict inequality, so we can delete these arcs and the node x from the graph $G_i(\theta'_{-i})$.

To find a node x in $G_i(\theta'_{-i})$ with $\mathrm{Adj}^+(x) = \emptyset$, we use depth-first search (DFS) starting with an arbitrary node in the graph. Doing so, we either find a node with no outgoing arcs, or we discover a directed cycle. In the first case, we lower the payment of the node as in the procedure described above and continue the DFS-procedure at the node considered before as long as there are still nodes remaining in the graph. In the case that we find a directed cycle C, all Inequalities (15) on C are fulfilled with equality and adding them up yields

$$0 = \sum_{(x_k, x_l) \in C} \left(P_i^{\theta'_{-i}}(x_l) - P_i^{\theta'_{-i}}(x_k) \right) = \sum_{(x_k, x_l) \in C} c_i^{\theta'_{-i}}(x_k, x_l), \tag{17}$$

where the first equality follows since C is a cycle. On the other hand, if the strict Inequality (16) corresponding to i and θ'_{-i} was fulfilled for any arc on C, we would obtain

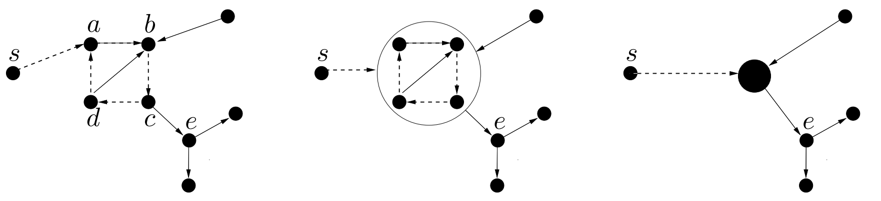
$$0 = \sum_{(x_k, x_l) \in C} \left(P_i^{\theta'_{-i}}(x_l) - P_i^{\theta'_{-i}}(x_k) \right) < \sum_{(x_k, x_l) \in C} c_i^{\theta'_{-i}}(x_k, x_l),$$

contradicting (17). Hence, the strict Inequality (16) cannot be satisfied for any arc on the cycle C. In this case, we contract all nodes on C to a single supernode and continue the DFS-procedure at this new node (see Figure 1).

When the above procedure terminates, every arc (x, x') in the complete, directed graph on $W(\theta'_{-i})$ is either contained in a cycle of arcs for which the corresponding Inequalities (15) are fulfilled with equality (so the Inequality (15) of (x, x') cannot be made strict in this graph), or the inequality corresponding to (x, x') is fulfilled with strict inequality.

Algorithm 1 summarizes the above discussion. For each agent i, the algorithm first calculates the set A of pairs $(\theta_i, x') \in \Theta_i \times X$ for which some strict Inequality (16) must hold. Then, for every possible bid vector $\theta'_{-i} \in \Theta_{-i}$ of the other agents, it calculates the set $W(\theta'_{-i})$ and a node potential in the complete, directed graph on $W(\theta'_{-i})$ via the Bellman-Ford Algorithm and perturbates this node potential such that each arc (x, x') is either contained in a cycle of arcs for which the corresponding Inequalities (15) are fulfilled with equality, or the inequality corresponding to (x, x') is fulfilled with strict inequality. The pairs $(\theta_i, x') \in \Theta_i \times X$ for which some strict Inequality (16) holds are then deleted from A, and the algorithm continues with the next bid vector $\theta'_{-i} \in \Theta_{-i}$ of the other agents. If the set A is still nonempty after processing all possible vectors $\theta'_{-i} \in \Theta_{-i}$, the remaining pairs $(\theta_i, x') \in A$ are pairs for which the second condition of the system cannot be satisfied, so the system does not have a solution. Otherwise, the algorithm continues with the next agent i. As a node in a graph can represent several outcomes due to previous contractions, the outcomes corresponding to each node u are stored in a set Outcomes(u).

The DFS-procedure used to find a node x in $G_i(\theta'_{-i})$ with $\text{Adj}^+(x) = \emptyset$ is implemented in the procedure PROCESS-GRAPH (Algorithm 2). $\pi(v)$ and color(v) denote the predecessor and the current state (GRAY = already visited, WHITE = not yet visited) of a node v, respectively. A stack-like data structure S is used to store the nodes to be processed next. It supports the operations FIRST(S) (returns first element of S), PUSH_FRONT(S, v) (inserts v as first element of S), and REMOVE(S, v) (deletes v from S). More details on depth-first can be found in Cormen et al. [1].



(a) DFS finds a cycle. (b) Contraction of the cycle (c) Graph after contraction

Figure 1: Contraction of a cycle to a supernode

Algorithm 1.

```
1: for all $i \in N$ do
2:     for all $\theta'_{-i} \in \Theta_{-i}$, $\theta_i \in \Theta_i$, $x' \in X$ do
3:         $\underline{c}_i^{\theta'_{-i}}(\theta_i, x') := \min_{\theta_{-i} \in \Theta_{-i}} \big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(x', \theta)\big)$
4:         $\overline{c}_i^{\theta'_{-i}}(\theta_i, x') := \max_{\theta_{-i} \in \Theta_{-i}} \big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(x', \theta)\big)$
5:     end for
6:     //Calculate the sets $C(\theta_i, x')$
```

```
7:  for all $\theta'_{-i} \in \Theta_{-i}$ do
8:      $K^{\theta'_{-i}}(x) := \{\theta_i \in \Theta_i : f(\theta_i, \theta'_{-i}) = x\}$ for $x \in X$.
9:      $W(\theta'_{-i}) := \{x \in X : K^{\theta'_{-i}}(x) \neq \emptyset\}$
10:     for all $\theta_i \in \Theta_i, x' \in W(\theta'_{-i})$ do
11:         if $f(\theta_i, \theta'_{-i}) \neq x'$ then
12:             $C(\theta_i, x') := C(\theta_i, x') \cup \{\theta'_{-i}\}$
13:         end if
14:     end for
15: end for
16: //Find the set $A$ of pairs $(\theta_i, x') \in \Theta_i \times X$ for which one inequality must be strict
17: $A := \Theta_i \times X$
18: for all $\theta'_{-i} \in \Theta_{-i}$ do
19:     for all $(x, x') \in W(\theta'_{-i}) \times W(\theta'_{-i})$ do
20:         $c_i^{\theta'_{-i}}(x, x') := \min_{\theta_i \in K^{\theta'_{-i}}(x)} \underline{c}_i^{\theta'_{-i}}(\theta_i, x')$
21:     end for
22:     for all $\theta_i \in \Theta_i$, $x' \in X$ do
23:         if $C(\theta_i, x') = \emptyset$, $\underline{c}_i^{\theta'_{-i}}(\theta_i, x') \neq \overline{c}_i^{\theta'_{-i}}(\theta_i, x')$, or $c_i^{\theta'_{-i}}(f(\theta_i, \theta'_{-i}), x') \neq \underline{c}_i^{\theta'_{-i}}(\theta_i, x')$
            then
24:             $A := A \setminus \{(\theta_i, x')\}$
25:         end if
26:     end for
27: end for
28: for all $\theta'_{-i} \in \Theta_{-i}$ do
29:     Choose $x \in W(\theta'_{-i})$ arbitrarily.
30:     Apply the Bellman-Ford Algorithm to the complete, directed graph $G$ with node
        set $W := W(\theta'_{-i})$, arc costs $c(x, x') := c_i^{\theta'_{-i}}(x, x')$, and start node $x$.
31:     if $G$ contains a negative cycle then
32:         STOP: $f$ cannot be implemented at all.
33:     else
34:         Denote the node potential obtained by the Bellman-Ford Algorithm by $P$.
35:         $V(G) := \emptyset$, $E(G) := \emptyset$
36:         for all $(x, x') \in W \times W$ with $x \neq x'$ do
37:             if $P(x') - P(x) = c(x, x')$ then
38:                 $V(G) := V(G) \cup \{x, x'\}$
39:                 $E(G) := E(G) \cup \{(x, x')\}$
40:             end if
41:         end for
42:         $G := (V(G), E(G))$
43:         PROCESS-GRAPH$(G, W, c, P)$
44:     end if
45:     for all $(x, x') \in W \times W$ with $x \neq x'$ do
46:         if $P(x') - P(x) < c(x, x')$ then
47:             for all $\theta_i \in K^{\theta'_{-i}}(x)$ do
48:                 $A := A \setminus \{(\theta_i, x')\}$
49:             end for
50:         end if
51:     end for
52:     for all $\theta'_i \in \Theta_i$ do
53:         $P_i(\theta'_i, \theta'_{-i}) := P(f(\theta'_i, \theta'_{-i}))$
```

```
54:         end for
55:     end for
56:     if $A \neq \emptyset$ then
57:         STOP: No payments $P_i(\theta')$ exist such that $\Gamma_{(f,P)}$ strongly implements $f$.
58:     end if
59: end for
60: STOP: The mechanism $\Gamma_{(f,P)}$ with payments $P_i(\theta')$ strongly implements $f$.
```

Algorithm 2. PROCESS-GRAPH(G, W, c, P)

```
 1: for all $u \in V(G)$ do
 2:     color$(u)$ := WHITE
 3:     $\pi(u)$ := NIL
 4:     Outcomes$(u) := \{u\}$
 5: end for
 6: while $w \in V(G)$ exists do
 7:     $S := \{w\}$
 8:     while $S$ not empty do
 9:         $u$ =FIRST$(S)$
10:         color$(u)$ := GRAY
11:         if Adj$^+(u) = \emptyset$ then
12:             PERTURBATE$(u, W, S, c, P)$
13:         else
14:             for all $v \in$ Adj$^+(u)$ do
15:                 if color$(v)$ = WHITE then
16:                     $\pi(v) := u$
17:                     PUSH_FRONT$(S, v)$
18:                 end if
19:             end for
20:             for all $v \in$ Adj$^+(u)$ do
21:                 if color$(v)$ = GRAY then
22:                     CONTRACT$(u, v, S)$
23:                     break
24:                 end if
25:             end for
26:         end if
27:     end while
28: end while
```

Algorithm 3. PERTURBATE(u, W, S, c, P)

```
1: $\epsilon := \min_{x \in \text{ Outcomes}(u)} \min_{x' \in W} c(x, x') - P(x') + P(x)$
2: for all  $x \in$ Outcomes$(u)$ do
3:     $P(x) := P(x) - \epsilon/2$
4: end for
5: for all $w \in$ Adj$^-(u)$ do
6:     Adj$^+(w)$ := Adj$^+(w) \setminus \{u\}$
7: end for
8: $V(G) := V(G) \setminus \{u\}$
9: REMOVE$(S, u)$
```

Algorithm 4. CONTRACT(u, v, S)

```
1:  $C := \{v\}$
2:  $w := u$
3:  **repeat**
4:      $C := C \cup \{w\}$
5:      $w := \pi(w)$
6:  **until** $w = v$
7:  //*Introduce new supernode* $v_C$
8:  $V(G) := V(G) \cup \{v_C\}$
9:  color$(v_C)$ := WHITE
10: $\pi(v_C) := \pi(v)$
11: Outcomes$(v_C) := \emptyset$
12: **for all** $w \in C$ **do**
13:     Outcomes$(v_C)$ := Outcomes$(v_C)$ $\cup$ Outcomes$(w)$
14:     $\mathrm{Adj}^+(v_C) := \mathrm{Adj}^+(v_C) \cup (\mathrm{Adj}^+(w) \setminus C)$
15:     $\mathrm{Adj}^-(v_C) := \mathrm{Adj}^-(v_C) \cup (\mathrm{Adj}^-(w) \setminus C)$
16:     REMOVE$(S, w)$
17: **end for**
18: **for all** $w \in \mathrm{Adj}^+(v_C)$ **do**
19:     $\mathrm{Adj}^-(w) := \mathrm{Adj}^-(w) \cup \{v_C\}$
20:     $\mathrm{Adj}^-(w) := \mathrm{Adj}^-(w) \setminus C$
21: **end for**
22: **for all** $w \in \mathrm{Adj}^-(v_C)$ **do**
23:     $\mathrm{Adj}^+(w) := \mathrm{Adj}^+(w) \cup \{v_C\}$
24:     $\mathrm{Adj}^+(w) := \mathrm{Adj}^+(w) \setminus C$
25: **end for**
26: $V(G) := V(G) \setminus C$
27: PUSH_FRONT $(S, v_C)$
```

Theorem 1. *Algorithm 1 correctly computes the payments P of a mechanism $\Gamma_{(f,P)}$ that strongly implements the given social choice function f in dominant strategies or decides that no such payments exist. The algorithm runs in time $\mathcal{O}(n \cdot |\Theta| \cdot |X|^3)$.*

Proof. For every fixed $\theta'_{-i} \in \Theta_{-i}$, the arc costs $c(x, x')$ calculated in the algorithm are given by

$$
\begin{aligned}
c_i^{\theta'_{-i}}(x, x') &= \min_{\theta_i \in K^{\theta'_{-i}}(x)} \underline{c}_i^{\theta'_{-i}}(\theta_i, x') \\
&= \min_{\theta_i \in K^{\theta'_{-i}}(x)} \min_{\theta_{-i} \in \Theta_{-i}} \Big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(x', \theta) \Big) \\
&= \min_{\substack{\theta_i, \theta'_i \in \Theta_i: \\ f(\theta_i, \theta'_{-i}) = x \\ f(\theta'_i, \theta'_{-i}) = x'}} \min_{\theta_{-i} \in \Theta_{-i}} \Big(V_i(f(\theta_i, \theta'_{-i}), \theta) - V_i(f(\theta'_i, \theta'_{-i}), \theta) \Big),
\end{aligned}
$$

which equals the arc costs $c_i^{\theta'_{-i}}(x, x')$ used in the discussion above. Hence, correctness of the algorithm follows from the arguments preceding the algorithm.

The running time can be estimated as follows: For each agent i, the calculation of the sets $C(\theta_i, x')$ and the set A needs time at most $\mathcal{O}(|\Theta_{-i}| \cdot |\Theta_i| \cdot |X|^2) = \mathcal{O}(|\Theta| \cdot |X|^2)$. In the for-loop starting in Line 28, the application of the Bellman-Ford Algorithm to the graph G in Line 30 needs time $\mathcal{O}(|W|^3) \leq \mathcal{O}(|X|^3)$. In the procedure PROCESS-GRAPH(G, W, c, P) in Line 43, at most $|W| \leq |X|$ contraction or perturbation steps are made since each such step reduces the number of nodes in the graph by at least one. Each call of the procedure CONTRACT(u, v, S) needs time at most $|W|^2 \leq |X|^2$ since each adjacency list and each set Outcomes$(\cdot)$ contains at most $|W|$ nodes and there are at most $|W|$ nodes in the cycle C. Each call of the procedure PERTURBATE(u, W, c, P) needs time at most $|W|^2 \leq |X|^2$ as well. Thus, the overall time needed for the procedure PROCESS-GRAPH(G, W, c, P) is at most $\mathcal{O}(|X|^3)$. Since every iteration of the loop except for the Bellman-Ford Algorithm and the call of PROCESS-GRAPH(G, W, c, P) needs time at most $\mathcal{O}(|\Theta_i| \cdot |X|^2)$, this implies that the overall time required for the for-loop starting in Line 28 is at most $\mathcal{O}(|\Theta_{-i}| \cdot |\Theta_i| \cdot |X|^3) = \mathcal{O}(|\Theta| \cdot |X|^3)$. Since there are n agents, we obtain an overall running time of $n \cdot \mathcal{O}(|\Theta| \cdot |X|^2) + n \cdot \mathcal{O}(|\Theta| \cdot |X|^3) = \mathcal{O}(n \cdot |\Theta| \cdot |X|^3)$ as claimed. $\square$

As already shown, the Weak Implementability Problem can be solved in the same way by just leaving out the steps needed to make sure that the strict inequalities in our system are fulfilled. Hence, the resulting algorithm solves the Weak Implementability Problem in time $\mathcal{O}(n \cdot |\Theta| \cdot |X|^3)$.

References

[1] T. H. Cormen, C. Stein, C. E. Leiserson, and R. L. Rivest: *Introduction to Algorithms.* 3rd edition. MIT Press (2009).

[2] H. Gui, R. Müller, and R. V. Vohra: *Dominant strategy mechanisms with multidimensional types.* In: *Computing and Markets* (2005).

[3] S. O. Krumke and C. Thielen: *Strong implementation of social choice function in dominant strategies.* COMSOC 2010.

[4] A. Mas-Colell, M. D. Whinston, and J. R. Green: *Microeconomic Theory.* Oxford University Press (1995).

[5] D. Mookherjee and S. Reichelstein: *Implementation via augmented revelation mechanisms.* Review of Economic Studies 57(3): pp. 453–475 (1990).

[6] J.-C. Rochet: *A necessary and sufficient condition for rationalizability in a quasilinear context.* Journal of Mathematical Economics 16: pp. 191–200 (1987).

[7] A. Schrijver: *Combinatorial Optimization*, volume 24 of *Algorithms and Combinatorics.* Springer (2003).

Clemens Thielen and Stephan Westphal
Department of Mathematics
University of Kaiserslautern
Paul-Ehrlich-Str. 14
67663 Kaiserslautern, Germany
Email: `{thielen,westphal}@mathematik.uni-kl.de`