# Mechanism Design for Software Agents with Complete Information ⋆

Thomas C. O'Connell [a,*,1] Richard E. Stearns [b]

[a] *Department of Mathematics and Computer Science, Skidmore College, Saratoga Springs, NY 12866*

[b] *Department of Computer Science, University at Albany, SUNY, Albany, NY 12222*

**Abstract**

We investigate the mechanism design problem when the agents and the mechanism have computational restrictions. In particular, we examine how results in the mechanism design literature are affected when the social choice rule requires the mechanism to solve a computationally difficult optimization problem. Both dominant strategy and Nash implementation are considered for a multiagent version of the Maximum Satisfiability problem. We show that the best a mechanism can guarantee is that at least half of the maximum number of simultaneously satisfiable agents will be satisfied by the outcome. Our analysis highlights some of the difficulties that arise in applying results from mechanism design to computational problems. In particular, our results show that using approximation in multiagent settings can be much less successful than in traditional computational settings because of the game theoretic guarantees required of the outcomes.

*Key words:* Algorithmic mechanism design, approximation algorithms, game theory, implementation theory
*PACS:*

⋆ A preliminary version of this paper appeared without many of the proofs in O'Connell and Stearns (2002). Material from that paper is used with the permission of Kluwer Academic Publishers.

* Corresponding author
*Email address:* oconnellT@acm.org (Thomas C. O'Connell).

[1] Supported in part by National Science Foundation Grant CCR-97-34936

## 1 Introduction

With the advent of Internet computing and electronic commerce, there has been increasing interest in computational systems, referred to as *multiagent systems*, that involve the interaction of many different computer programs. These programs, or *software agents*, may be written by different people or companies with different goals in mind. In other words, the programs can be viewed as self-interested. Not surprisingly, the design and analysis of multiagent systems involves the tools of game theory and mechanism design (see [32], [34], [18] and [12] for examples). Developing a clear understanding of the computational issues involved in mechanism design should facilitate its use in multiagent system design. Therefore, in this paper, we consider how placing computational limitations on the agents and the mechanism affects classic results in the mechanism design literature. In particular, we investigate the effect of restricting the agents and the mechanism to polynomial time computation. (Section 3 provides details on exactly how this is done.) To focus our investigation, we consider a particular problem which we call *Multiagent MAXSAT* and restrict ourselves to complete information environments. In Multiagent MAXSAT, each agent's preferences over the set of possible outcomes can be described by a disjunction over negated and unnegated Boolean variables. The problem of assigning truth values to a set of variables so that the number of satisfied disjunctions is maximized is known to be a computationally difficult problem (see [6]). According to the widely held belief of computer scientists and logicians, namely that P $\neq$ NP, it would be impossible for the mechanism to always maximize the number of satisfied agents if the agents and the mechanism are limited to polynomial time computation. Therefore, any polynomial time mechanism must settle for outcomes that are approximately optimal. (Readers unfamiliar with the P vs NP question should refer to the Appendix for an explanation.)

The main results of this paper are as follows:

(1) The revelation principle applies when the mechanism and the agents are restricted to polynomial time. It *does not* apply when the mechanism is restricted but the agents are not.
(2) We provide a mechanism with a non-dictatorial outcome function that

implements MAXSAT in dominant strategies. The mechanism runs in polynomial time but the agents require non-polynomial time to compute their dominant strategies. (Throughout this paper, we assume that P $\neq$ NP.)

(3) We provide a mechanism such that all dominant strategy equilibrium outcomes satisfy at least half of the agents. In this case, the mechanism and the agents use only polynomial time.

(4) We provide a polynomial time mechanism that guarantees that each Nash equilibrium outcome satisfies at least half of the agents.

(5) In the case of strong implementation in dominant strategy, Nash, undominated Nash or subgame perfect equilibrium, it is impossible to guarantee that the equilibrium outcomes will satisfy more than half of the maximum number of simulaneously satisfiable agents. In contrast, there are approximation algorithms for the non-multiagent version of MAXSAT that guarantee that 3/4 of the maximum number of simulaneously satisfiable agents will be satisfied. (See [39,8,1].)

The Multiagent MAXSAT problem is simplistic in that each agent is limited to preferences defined by simple unweighted disjunctions. However, since most of our results demonstrate the difficulty of designing mechanisms for such a restricted version of the problem, these difficulties will carry over to more realistic models.

*Related Work*

A computational formulation of the mechanism design problem is presented in [23]. (See also [25].) They studied dominant strategy implementation for a task scheduling problem in which there is a set of tasks to be distributed among a group of agents in such a way that the time at which the last task is completed is minimized. The agents, who prefer to do no work at all, have different times in which they can perform each task. These times are unknown to the mechanism. The task scheduling problem is a computationally difficult problem. Therefore, even if the agents truthfully reveal their times to the mechanism, a mechanism restricted to polynomial time computation cannot always find an optimal task distribution. The best the mechanism can hope for is to find an

approximately optimal task distribution. An "approximation" mechanism is provided in [23] such that each agent's unique dominant strategy is to truthfully inform the mechanism of their times. In [24], it is shown that, under rather mild assumptions of what constitutes a reasonable social choice rule, dominant strategy implementation of reasonable social choice rules that approximately maximize the sum of the agents' utilities is impossible. Restricted conditions under which such an approximation mechanism can be found are provided in [37]. Approximation in the context of combinatorial auctions is studied in [13].

All of the papers mentioned above consider only dominant strategy implementation whereas we consider both dominant strategy and Nash implementation. We also briefly consider undominated Nash and subgame perfect implementation. Furthermore, the work cited above studies only truthful revelation mechanisms. For these mechanisms, the agents have no computation to perform. They simply pass their preferences unchanged to the mechanism. In our work, the agents computation time plays a significant role in the results. (In [24], the agents' computational abilities are considered in trying to overcome the impossibility results.)

This paper can also be considered a contribution to the mechanism design literature on bounded rationality. In [19], it is pointed out that bounded rationality has received little attention in the mechanism design community. We believe that computer science provides a rich set of tools for modeling bounded rationality. Ideas from computer science have been used previously by game theorists and computer scientists studying bounded rationality in repeated games [21,22,3,30,36,28].

*Outline*

Section 2 provides a short review of mechanism design and formally defines the Multiagent MAXSAT problem as a computational problem. Section 3 formalizes the idea of a polynomial time mechanism. Section 4 discusses polynomial time revelation mechanisms. Sections 5 and 6 look at designing polynomial time mechanisms using dominant strategy and Nash equilibrium respectively for Multiagent MAXSAT. Section 7 provides a proof that the best a mech-

anism can guarantee is that half of the maximum number of simultaneously satisfiable agents will be satisfied. Since some readers may not be familiar with the relevant complexity concepts from computer science, we have included an appendix that briefly explains these concepts.

## 2 Motivation and Definitions

The mechanism design problem is intended to model situations in which a set of self-interested agents must come to a collective decision. The designer of the decision making process would like the decision to be good for the society as a whole where "good" is defined by some "social choice rule". The way the collective decision is made is that each agent sends a message to a decision making procedure. This decision making procedure then selects an outcome based on the messages sent by the agents and a predefined set of selection rules. Each agent, being self-interested, may prefer outcomes that are not socially desirable. Therefore, each agent will try to manipulate the decision making procedure into choosing outcomes that are better for the agent at the expense of the other agents. The "mechanism design problem" is the problem of designing the message sets and the selection rules so that, if the agents choose their messages rationally, the selected outcome is one of the outcomes defined by the social choice rule to be a good outcome. "Selecting rationally" means that the selected messages satisfy some equilibrium condition. The design problem is complicated by the fact that the agents are expected to select a message based on their own self-interest rather than a message most useful for computing an outcome best for society. For surveys on this subject, see [16,14,19,15]. Here we deal with the additional complication that, even if the agents' messages perfectly reflected their true preferences, the best outcome may be difficult to compute. In other words, even if the decision making procedure knew exactly what each agent wanted, it would be difficult to compute a good outcome.

**Definition 2.1** *A* **mechanism design problem** *consists of*

*(1) a finite set of* **outcomes***;*
*(2) a finite set of* **agents***;*

*(3) for each agent, a* **preference set** *which is a set of possible preference relations on the set of outcomes (a vector of preferences, one from each agent's set of preferences, is called a* **preference profile***);*

*(4) a* **social choice rule** *which specifies, for each preference profile, a non-empty set of outcomes which are considered desirable from a social standpoint.*

Intuitively, the preference sets represent the set of preferences that an agent might have. A preference profile represents the preferences that the agents actually do have. The social choice rule says which outcomes are best from society's viewpoint given a profile of the agents' true preferences. The concept of a "mechanism for a mechanism design problem" will be defined separately.

We next define a MAXSAT mechanism design problem for which the outcomes are truth assignments to a set of Boolean variables and the agents' preferences are determined by clauses. A clause is a disjunction of negated and unnegated Boolean variables. For a given clause, an agent prefers assignments which satisfy the clause to assignments which do not. The agent is indifferent among assignments which satisfy the clause and indifferent among assignments which do not. For example, if an agent's preference is determined by the clause $(x \vee y \vee \bar{z})$, the agent prefers assignments which assign $x$ or $y$ the value True or which assign $z$ the value False.

**Definition 2.2** *A* **MAXSAT mechanism design problem** *is a follows:*

*(1) the set of outcomes is the set of assignments to some specified finite set of Boolean variables;*

*(2) the agent set is any finite set;*

*(3) the preference set is specified by the set of clauses on the Boolean variables (and a preference profile is therefore specified by a vector of clauses);*

*(4) the social choice rule specifies that, for each preference profile $\theta$, any outcome that maximizes the number of satisfied clauses in $\theta$ is socially desirable.*

Notice that, to specify a MAXSAT mechanism design problem, one need only supply a list of variables and the number of agents. The preference sets and social choice rule are then implied by the definition.

**Definition 2.3** *A* **mechanism for a mechanism design problem** *consists of*

*(1) a message set for each agent;*
*(2) an* **outcome function** *which is a mapping from the set of message vectors (one component for each agent) to the set of outcomes.*

We also define a mechanism for a set of mechanism design problems:

**Definition 2.4** *A* **mechanism for a set** $S$ **of mechanism design problem** *consists of the following:*

*(1) for each mechanism design problem instance in* $S$*, a message set for each agent;*
*(2) an* **outcome function** *which maps a problem instance* $p \in S$ *and a message vector for* $p$ *(one component for each agent) to an outcome from the set of outcomes.*

Note that the outcome function of a "mechanism for a set of mechanism design problems" has two kinds of parameters: parameters describing an instance of the mechanism design problem and parameters for describing a message vector. It is the complexity of computing this outcome function that is the subject of this paper. Although most of the mechanism design literature is in fact about mechanisms for sets of mechanism design problem, it has usually been unnecessary for these papers to make such a careful distinction between mechanisms for particular mechanism design problem and mechanisms for a set of mechanism design problem.

When a mechanism is imposed on a mechanism design problem, the result is a kind of a "game". At the start of the game, each agent is given a preference from his preference set. In the *complete information* case studied here, the agents are also informed about the preferences of the other agents. The agents each send a message to a procedure which computes an outcome. The objective of the design problem is to design the mechanism so that (if agents play rationally) the computed outcome is an outcome preferred by the social choice rule.

The main topic of this paper is designing mechanisms for the set of MAXSAT

mechanism design problem. (We refer to this as the **Multiagent MAXSAT problem**.) In this case, the mechanism design problem description supplied to the output function consists of a list of Boolean variables and the number of agents. The number of agents is actually redundant information since the procedure for computing outcomes can infer this number from the number of components in the message vector.

Throughout the remainder of this paper, we use the term mechanism to refer to a mechanism for a set of mechanism design problems. This requires us to modify many of the standard definitions and results from the mechanism design literature so that they take into account the problem instance $p \in S$. In particular, we generalize the definition of a social choice rule to be a mapping of a mechanism design problem instance and a preference profile for that problem instance to a set of socially desirable outcomes. For example, for Multiagent MAXSAT, we define the social choice rule as follows:

**Definition 2.5** *For any problem instance $p$ in the set of MAXSAT mechanism design problems and any preference profile $\theta$ for $p$,*

$$\text{MAXSAT}(p, \theta) = \{t : t \textit{ is a truth assignment that satisfies } N^*(\theta) \textit{ clauses}\}$$

*where $N^*(\theta)$ is the maximum number of simultaneously satisfiable clauses in $\theta$.*

There are different degrees to which a mechanism can succeed in satisfying a social choice rule. These are defined as follows:

**Definition 2.6** *Let $F$ be a social choice rule for a set of mechanism design problems. Given a mechanism $\Gamma$, let $E(p, \theta)$ be the set of equilibrium outcomes for a problem instance $p$ and a preference profile $\theta$. We say:*

*(1)* $\Gamma$ **implements** *$F$ if $E(p, \theta) \cap F(p, \theta) \neq \emptyset$ for all $p$ and $\theta$.*
*(2)* $\Gamma$ **strongly implements** *$F$ if $\emptyset \neq E(p, \theta) \subseteq F(p, \theta)$ for all $p$ and $\theta$.*
*(3)* $\Gamma$ **fully implements** *$F$ if $E(p, \theta) = F(p, \theta)$ for all $p$ and $\theta$.*

If multiple equilibria exist, it might be difficult to argue that one equilibrium will be played while another will not since all equilibria are equally rational according to any particular equilibrium criterion. The definition of *implemen-*

*tation* does not exclude the possibility of undesirable equilibrium outcomes. *Strong implementation*[2] eliminates mechanisms that do not guarantee that every equilibrium outcome is socially desirable. *Full implementation* ensures that all socially desirable outcomes are achievable as equilibrium outcomes. In line with the philosophy of approximation algorithms, we want to guarantee that all outcomes reach some threshold of acceptability. We are not necessarily concerned with making all acceptable outcomes possible. Therefore, in designing mechanisms to control the interactions of a group of software agents, we consider the mechanism successful if it achieves strong implementation. However, we return to this discussion in Section 6 where we provide some justification for trying to achieve full implementation.

## 3 Polynomial Time Mechanisms

An algorithm is said to run in **polynomial time** if there is a polynomial $q$ such that, for every possible input, the algorithm produces an output in no more than $q(n)$ primitive computational steps, where $n$ is the size of the input. (For more details on this concept see [9], [6] or [5]). In the case of outcome functions and strategy functions for MAXSAT, "polynomial in the input size" is equivalent to "polynomial in the number of Boolean variables and the number of agents." The agents' strategy functions take as input the description of the mechanism design problem instance (i.e. the set of variables and the number of agents) and a clause for each agent. The number of literals in each clause is bounded by twice the number of variables. The outcome functions take the description of the mechanism design problem instance as input together with a vector of messages. Under the reasonable assumption that the message lengths are polynomial in the number of variables and the number of agents, the entire input is so bounded. This assumption about message lengths must hold if the messages were produced by strategy functions in polynomial time.

We consider implementation in dominant strategy and Nash equilibrium which are defined below. In these definitions and in the remainder of this paper, $v_i$

[2] In [16], this is called *weak implementation* while we use *strong implementation* following [14].

denotes the $i$-th component of a vector $v$ while $v_{-i}$ denotes the vector $v$ with the $i$-th component removed. We use $(v'_i, v_{-i})$ to represent the vector $v$ with the $i$-th component replaced by $v'_i$.

Let $g(p, m)$ denote the outcome of a mechanism $\Gamma$ given the problem instance $p$ and message profile $m$. Let $t \succeq_i t'$ denote that agent $i$ prefers outcome $t$ to outcome $t'$.

**Definition 3.1** *A vector of strategy functions $s^*$ (referred to as a strategy profile) is a* **dominant strategy equilibrium** *of a mechanism $\Gamma$ if, for each agent $i$, all problem instances $p$, and all preference profiles $\theta$,*

$$g(p, (s_i^*(p, \theta), m_{-i})) \succeq_i g(p, (m'_i, m_{-i}))$$

*for all messages $m'_i$ and all message profiles $m_{-i}$.*

In other words, a strategy profile is a dominant strategy equilibrium if, no matter what the other agents do, no agent $i$ can benefit by sending a message different from the one prescribed by $s_i^*$.

**Definition 3.2** *A strategy profile $s^*$ is a* **Nash equilibrium** *of a mechanism $\Gamma$ if, for each agent $i$, all problem instances $p$, and all preference profiles $\theta$,*

$$g(p, (s_i^*(p, \theta), s_{-i}^*(p, \theta))) \succeq_i g(p, (m'_i, s_{-i}^*(p, \theta)))$$

*for all messages $m'_i$.*

In other words, a strategy profile is a Nash equilibrium if no agent can benefit by unilaterally deviating from the prescribed message.

**Definition 3.3** *An outcome $x$ is a* **polynomial time equilibrium outcome** *of a mechanism $\Gamma$ if there is a equilibrium strategy profile $s^*$, a problem instance $p$, and a preference profile $\theta$ such that $\Gamma$ outputs $x$ when the message profile is $s^*(p, \theta)$ and, for each agent $i$, $s_i$ is a polynomial time strategy function.*

We now define what it means for a mechanism to implement a social choice rule in polynomial time.

**Definition 3.4** *Let $F$ be a social choice rule. Let $\Gamma$ be a polynomial time mechanism. Let $\mathrm{PE}(p,\theta)$ be the set of polynomial time equilibrium outcomes of $\Gamma$ for problem instance $p$ and preference profile $\theta$. We say that,* **in polynomial time for polynomial time bounded agents**,

*(1)* $\Gamma$ **implements** $F$ *if* $\mathrm{PE}(p,\theta) \cap F(p,\theta) \neq \emptyset$ *for all* $p$ *and* $\theta$.
*(2)* $\Gamma$ **strongly implements** $F$ *if* $\emptyset \neq \mathrm{PE}(p,\theta) \subseteq F(p,\theta)$ *for all* $p$ *and* $\theta$.
*(3)* $\Gamma$ **fully implements** $F$ *if* $\mathrm{PE}(p,\theta) = F(p,\theta)$ *for all* $p$ *and* $\theta$.

The restriction to polynomial time equilibrium outcomes requires some discussion. As the following lemma shows, if there is at least one polynomial time equilibrium strategy profile, then every equilibrium outcome is a polynomial time equilibrium outcome:

**Lemma 3.1** *Given a mechanism $\Gamma$, let $PE(p,\theta)$ be the set of polynomial time equilibrium outcomes for problem instance $p$ and preference profile $\theta$. Let $E(p,\theta)$ be the set of all equilibrium outcomes for problem instance $p$ and preference profile $\theta$. If there is at least one polynomial time equilibrium strategy profile for $\Gamma$, then $PE(p,\theta) = E(p,\theta)$ for all $p$ and $\theta$.*

*Proof.* If suffices to show that $E(p,\theta) \subseteq PE(p,\theta)$ for all $p$ and $\theta$. Let $\hat{p}$ be any problem instance and $\hat{\theta}$ be any preference profile consistent with $\hat{p}$. Let $t$ be any member of $E(\hat{p},\hat{\theta})$. Let $\hat{s}$ be an equilibrium strategy profile such that $t$ is the outcome generated by $\Gamma$ when given $\hat{s}(\hat{p},\hat{\theta})$. Let $s^*$ be any polynomial time equilibrium strategy profile for $\Gamma$. Define a new strategy profile $s'$ such that for all $i$,

$$s'_i(p,\theta) = \begin{cases} \hat{s}_i(\hat{p},\hat{\theta}) \text{ if } p = \hat{p} \text{ and } \theta = \hat{\theta} \\ s^*_i(p,\theta) \text{ otherwise} \end{cases}$$

For each $i$, $s'_i$ is polynomial time computable since $\hat{s}_i(\hat{p},\hat{\theta})$ is fixed and $s^*_i$ is polynomial time computable. Furthermore, $s'$ is an equilibrium strategy profile since both $s^*$ and $\hat{s}$ are. Since $t$ is the outcome generated by $\Gamma$ when given $\hat{s}(\hat{p},\hat{\theta}) = s'(p,\theta)$ and $s'$ is a polynomial time strategy profile, $t \in PE(\hat{p},\hat{\theta})$. Therefore, $E(\hat{p},\hat{\theta}) \subseteq PE(\hat{p},\hat{\theta})$. ∎

As a result, we have the following proposition:

**Proposition 3.1** *If* $\Gamma$ *strongly [fully] implements a social choice rule* $F$ *in polynomial time for polynomial time bounded agents then* $\Gamma$ *strongly [fully] implements* $F$*.*

*Proof.* Since $\Gamma$ strongly implements $F$ in polynomial time for polynomial time bounded agents, $\Gamma$ has some polynomial time equilibrium strategy profile. The result then follows from Lemma 3.1. ∎

In Section 4, we show that MAXSAT is not implementable in polynomial time for polynomial time bounded agents. Therefore, the best one can hope for is to find some approximation to MAXSAT that is implementable in polynomial time for polynomial time bounded agents. An approximation to MAXSAT is defined as a social choice rule of the form:

$$c\text{-MAXSAT}(p, \theta) = \{t : t \text{ satisfies at least } cN^*(\theta) \text{ clauses}\}$$

where $c \in (0, 1)$ and $N^*(\theta)$ is the maximum number of simultaneously satisfiable clauses in $\theta$. We say that MAXSAT is *approximately implementable* if there is some constant $c \in (0, 1)$ such that $c$-MAXSAT is implementable. In Sections 5 and 6, we show that $\frac{1}{2}$-MAXSAT is strongly implementable in dominant strategy and Nash equilibrium respectively. In Section 7, we show that $c = 1/2$ is the largest value for which $c$-MAXSAT is strongly implementable.

## 4 Revelation Mechanisms

Mechanisms which require the agents to declare their preferences (truthfully or falsely) form an important class of mechanisms known as *revelation mechanisms*. A social choice rule $F$ is said to be *truthfully implementable* if there is a revelation mechanism for which truthful preference declarations by all the agents constitutes an equilibrium with outcome in $F(p, \theta)$ for all problem instances $p$ and all preference profiles $\theta$. The following result is known as the *Revelation Principle*. This principle applies to many equilibrium concepts including dominant strategy and Nash equilibrium so we state it without specifying a particular equilibrium concept (see the discussion in [15] on p. 182-183):

**Proposition 4.1 (The Revelation Principle)** *Suppose there exists a mechanism $\Gamma$ that implements social choice rule $F$. Then $F$ is truthfully implementable.*

*Proof.* See for example [14]. ∎

The Revelation Principle has a polynomial time analog if the agents are restricted to polynomial time strategies. This result holds for any equilibrium concept for which the standard Revelation Principle applies.

**Theorem 4.1 (The Polynomial Time Revelation Principle)** *If a social choice rule $F$ is implementable in polynomial time for polynomial time bounded agents then $F$ is truthfully implementable in polynomial time for polynomial time bounded agents.*

*Proof.* In the standard proof of the revelation principle (see [14] for example), we let $\Gamma$ be a mechanism that implements $F$ and $s^*$ be any equilibrium strategy profile that results in an outcome in $F(p, \theta)$ for all $p$ and $\theta$. Then a direct revelation mechanism $\Gamma^*$ is created which, given a problem instance and a declared preference profile $\hat{\theta}$, simulates $\Gamma$ on $s^*(p, \hat{\theta})$. It is straighforward to show that truthtelling is an equilibrium strategy for $\Gamma^*$.

The proof of the Polynomial Time Revelation Principle follows immediately from the standard proof since, if $\Gamma$ and $s^*$ are computable in polynomial time, so is $\Gamma^*$. ∎

As the following proposition shows, MAXSAT is not truthfully implementable by a polynomial time revelation mechanism. Combined with Theorem 4.1, this implies that MAXSAT is not polynomial time implementable for polynomial time bounded agents.

**Proposition 4.2** MAXSAT *is not truthfully implementable by a polynomial time revelation mechanism.*

*Proof.* Assuming P $\neq$ NP, there is no algorithm that can find a member of MAXSAT$(p, \theta)$ in polynomial time for all $p$ and $\theta$. Suppose MAXSAT is truthfully implemented by a mechanism $\Gamma$. Then the polynomial time algo-

rithm that computes $\Gamma$ finds a member of MAXSAT$(p, \theta)$ in polynomial time for all $p$ and $\theta$ contradicting our assumption that P $\neq$ NP. ∎

**Corollary 4.1** MAXSAT *is not polynomial time implementable for polynomial time bounded agents.*

*Proof.* Immediate from Proposition 4.2 and Theorem 4.1. ∎

As the following propositions shows, if the agents are not restricted to polynomial time, MAXSAT is Nash implementable by a polynomial time mechanism. This implies that the revelation principle does not apply when the mechanism is restricted to polynomial time but the agents are unrestricted.

**Proposition 4.3** *If the agents are unrestricted then* MAXSAT *is Nash implementable by a polynomial time mechanism.*

*Proof.* We need to show that there is a polynomial time mechanism that Nash implements MAXSAT when the agents have no computational restrictions. In this mechanism, each agent $i$ will declare a clause $\theta_i$ and propose an outcome $t_i$. The mechanism will select as the outcome the proposed outcome that satisfies the most declared clauses. If there is a tie, the mechanism will take the proposed outcome of the least numbered agent involved in the tie. We claim that the strategy profile where each agent $i$ declares its true clause and proposes any outcome that satisfies itself and the maximum number of other agents is a Nash equilibrium. Note that the agents' strategies are not computable in polynomial time since they require each agent to solve the (non-multiagent) MAXSAT problem.

We need to show that no agent has incentive to deviate from this strategy. For any $t$ and $\hat{\theta}$, let $N(\hat{\theta}, t)$ be the number of clauses in $\hat{\theta}$ that are satisfied by outcome $t$. First note that agent $i$ would have no reason to deviate if it is satisfied by the selected outcome. Let $\theta$ be the preference profile. Let $t_j$ be the outcome selected when each agent follows the strategy described above. Assume agent $i$ is not satisfied by $t_j$. The number of agents satisfied by agent $i$'s proposed outcome, $t_i$, is either less than the number satisfied by $t_j$ or $t_i$ and $t_j$ satisfy the same number of agents and $j < i$. In other words, $N(\theta, t_i) \leq N(\theta, t_j)$ with strict inequality if $j > i$. Since $t_i$ satisfies agent $i$

and the maximum number of other agents, $N((\theta_i', \theta_{-i}), t) \leq N(\theta, t_i)$ for all $\theta_i'$ and all $t$ such that $t$ satisfies $\theta_i$. Furthermore, since $t_j$ does not satisfy $\theta_i$, $N(\theta, t_j) \leq N((\theta_i', \theta_{-i}), t_j)$ for all $\theta_i'$. Combining these three inequalities, we have $N((\theta_i', \theta_{-i}), t) \leq N((\theta_i', \theta_{-i}), t_j)$ for all $\theta_i'$ and all $t$ such that $t$ satisfies $\theta_i$ with strict inequality if $j > i$. Hence, agent $i$ cannot change the selected outcome from an outcome that he is not satisfied with to an outcome that he is satisfied with by choosing a different message. Therefore, agent $i$ cannot benefit by deviating from the strategy. In other words, the strategy profile is a Nash equilibrium.

Notice that with this strategy profile, at least one of the agents proposes an outcome that is in MAXSAT$(p, \theta)$. Therefore, the outcome selected by the mechanism will be in MAXSAT$(p, \theta)$ which implies the mechanism implements MAXSAT. Furthermore, the computation performed by the mechanism is trivial. It simply needs to calculate and compare the number of clauses satisfied by each proposed outcome. This can clearly be done in polynomial time. ∎

In the proof of Proposition 4.3, we see that, if the mechanism is computationally restricted but the agents are not, then the mechanism can be designed in such a way that the agents perform computation on the part of the mechanism. In particular, the agents are being forced to solve a version of the maximum satisfiability problem and provide the result of their computation to the mechanism. The proof would also work in an incomplete information environment if the mechanism included two stages. In the first stage, the agents would declare their preferences to each other and the mechanism. In the second stage, the agents would propose an outcome. By lying in the first stage, an agent could affect the outcomes proposed by the other agents but a similar argument to the one above shows that the agents cannot affect the proposed outcomes in a beneficial way.

## 5 Dominant Strategy Implementation

When dealing with dominant strategy implementation, one must contend with an impossibility result known as the Gibbard-Satterthwaite Theorem ([7,35]) which restricts the set of implementable social choice functions to those that

are dictatorial. (A social choice function maps a problem instance and a preference profile to a single outcome.)

**Definition 5.1** *A social choice function $f$ is* **dictatorial for problem instance** *$p$ if there is a single agent $i$ such that, for all preference profiles $\theta$, $f(p, \theta)$ is agent $i$'s most preferred outcome.*

We say a social choice function is *truthfully implementable at $p$* if there is a revelation mechanism that implements $f$ and for which there is an equilibrium strategy profile $s^*$ with $s^*(p, \theta) = \theta$.

**Proposition 5.1 (The Gibbard-Satterthwaite Theorem)** *Let $f$ be any social choice function. Suppose, for problem instance p, the set of possible outcomes $X$ is finite and contains at least three elements and that the range of $f$ restricted to $p$ is $X$. Further suppose that the set of possible preference relations over $X$ contains the set of strict preferences over $X$. Then $f$ is truthfully implementable at $p$ in dominant strategies if and only if $f$ is dictatorial for problem instance $p$.*

Because of the Gibbard-Satterthwaite Theorem, dominant strategy implementation is generally studied in *quasilinear* environments. In a quasilinear environment, the agents' have some sort of transferrable good, i.e. money, and the outcomes include the transfer of money to or from individual agents. An agent's utility of an outcome is then defined to be the value the agent assigns to the non-monetary part of the outcome plus the actual amount of money transferred to the agent.

Since the set of preference relations for multiagent MAXSAT does not include the set of strict preference relations, the Gibbard-Satterthwaite Theorem does not apply to multiagent MAXSAT. Strict preferences are not possible because, if an agent's clause does not include a variable $x_i$, the agent is indifferent between truth assignments that are identical except in their assignment to $x_i$. Furthermore, if the agent's clause includes more than one literal, then the agent is indifferent between truth assignments that satisfy at least one of the literals.

Since the Gibbard-Satterthwaite Theorem does not apply, we are free to consider environments that are not quasilinear. As the following proposition

shows, if there are no computational restrictions on the mechanism, MAXSAT is truthfully implementable in dominant strategies using an outcome function that is not dictatorial for any problem instance.

**Proposition 5.2** MAXSAT *is truthfully implementable in dominant strategies by a revelation mechanism with an outcome function that is not dictatorial for any problem instance $p$.*

*Proof.* Order the truth assignments lexicographically where $x_i = \text{True}$ comes before $x_i = \text{False}$ for $1 \leq i \leq n$. Define a revelation mechanism $\Gamma$ that chooses the first truth assignment in the lexicographic ordering that satisfies the maximum number of simultaneously satisfiable clauses in the declared preference profile. For any truth assignment $\hat{t}$ and any preference profile $\hat{\theta}$, define $N(\hat{\theta}, \hat{t})$ to be the number of clauses in $\hat{\theta}$ that $\hat{t}$ satisfies.

Let $p$ be any problem instance. Fix $i$ and let $\theta_i$ be agent $i$'s true clause. To prove that truth telling is a dominant strategy for agent $i$, we need to show that no matter what preferences the other agents declare, agent $i$ can do no better than to declare its true preference $\theta_i$.

Let $\hat{\theta}_{-i}$ be any preference profile declared by the other agents. Let $t$ be the outcome when the message profile is $(\theta_i, \hat{\theta}_{-i})$. Suppose agent $i$ is not satisfied by $t$. Let $t'$ be any truth assignment that does satisfy $\theta_i$. Let $\hat{\theta}_i$ be any clause that agent $i$ can declare.

Since $t'$ satisfies $\theta_i$, $N((\hat{\theta}_i, \hat{\theta}_{-i}), t') \leq N((\theta_i, \hat{\theta}_{-i}), t')$. Since $t$ does not satisfy $\theta_i$, $N((\theta_i, \hat{\theta}_{-i}), t) \leq N((\hat{\theta}_i, \hat{\theta}_{-i}), t)$. But since $t$ is chosen over $t'$ when the agents declare $(\theta_i, \hat{\theta}_{-i})$, it must be the case that $N((\theta_i, \hat{\theta}_{-i}), t') \leq N((\theta_i, \hat{\theta}_{-i}), t)$. Combining these inequalities we have, $N((\hat{\theta}_i, \hat{\theta}_{-i}), t') \leq N((\hat{\theta}_i, \hat{\theta}_{-i}), t)$. Furthermore, if $N((\hat{\theta}_i, \hat{\theta}_{-i}), t') = N((\hat{\theta}_i, \hat{\theta}_{-i}), t)$, we must have $N((\theta_i, \hat{\theta}_{-i}), t') = N((\theta_i, \hat{\theta}_{-i}), t)$ and, therefore, the tie breaking rules must favor $t$, i.e., $t$ comes before $t'$ in the ordering. Thus, regardless of what clause agent $i$ declares, $t'$ will not be the outcome chosen by the mechanism. Since this is true for any $t'$ that satisfies agent $i$ and any $\hat{\theta}_{-i}$, agent $i$ cannot improve the outcome for himself by lying. Hence, truth telling is a dominant strategy.

To see that the outcome function is not dictatorial for $p$, suppose the preference

profile is $(\bar{x}_1, x_1, \ldots, x_1)$, i.e. agent 1 is satisfied only when $x_1 =$ False and the other agents are satisfied only when $x_1 =$ True. The mechanism sets $x_1$ to True so the outcome is not agent 1's most preferred outcome. The same argument can be applied to the other agents which implies that the outcome chosen by the mechanism is not the same agent's most preferred outcome for every $\theta$. ∎

*Approximation Mechanisms for Dominant Strategy Implementation*

We know from Corollary 4.1 that MAXSAT cannot be implemented in polynomial time so we are interested in determining whether there is some constant $c$ such that $c$-MAXSAT can be implemented in polynomial time. The question is, can we convert one of the many existing approximation algorithms for non-multiagent version of MAXSAT into a mechanism that truthfully implements $c$-MAXSAT? In [26], we show that a (1/2)-approximation algorithm and a (2/3)-approximation algorithm from [11] do not result in truthful mechanisms. It is possible to strongly implement $\frac{1}{2}$-MAXSAT in polynomial time, however, using an mechanism that we call the *complement mechanism* which is based on the following property of MAXSAT .

**Lemma 5.1** *For any truth assignment $t$, let $\bar{t}$ denote the truth assignment such that for every variable $v$, $\bar{t}(v) =$ True if and only if $t(v) =$ False. For all truth assignments $t$, either $t$ or $\bar{t}$ satisfies at least half of the maximum number of simultaneously satisfiable clauses.*

*Proof.* Let $\theta_i$ be any clause that $t$ does not satisfy. Let $l_i$ be a literal in $\theta_i$. Then $t(l_i) =$ False which implies $\bar{t}(l_i) =$ True. Therefore, $\bar{t}$ satisfies $\theta_i$. Hence, for every clause $\theta_i$, either $t$ or $\bar{t}$ must satisfy $\theta_i$, which implies that one of them must satisfy at least half the total number of clauses. ∎

We can use this property to develop a polynomial time mechanism that strongly implements $\frac{1}{2}$-MAXSAT. For each problem instance $p$, fix a truth assignment $t_p$ and define a social choice function $f$ as follows:

$$f(p, \theta) = \begin{cases} \bar{t}_p \text{ if } \bar{t}_p \text{ satisfies more clauses in } \theta \text{ than } t_p \\ t_p \text{ otherwise} \end{cases} \quad (1)$$

This social choice function is not dictatorial since, for any problem instance $p$ and for any agent $i$, we can define $\theta$ such that $t_p$ does not satisfy $\theta_i$ but does satisfy every clause in $\theta_{-i}$ while $\bar{t}_p$ does not satisfy any clause in $\theta_{-i}$. For example, let $t_p = \bar{x}_1 x_2 x_3$ and $\theta = (x_1, x_2, x_3)$. We have $f(p, \theta) = t_p$ which is not agent 1's most preferred outcome. A similar argument applies to the other two agents which implies that $f(p, \theta)$ is not the same agent's most preferred outcome for every $\theta$. Lemma 5.1 implies that $f(p, \theta) \in \frac{1}{2}$-MAXSAT$(p, \theta)$ for all $p$ and $\theta$. Therefore, any mechanism that truthfully implements $f$, also truthfully implements $\frac{1}{2}$-MAXSAT.

Define the following revelation mechanism:

**Mechanism 5.1**

*Given a problem instance p:*

*1. Let $t_p$ be the fixed truth assignment from the definition of $f$ in Eq. (1).*

*2. If $t_p$ satisfies at least as many of the declared clauses as $\bar{t}_p$*

*3. then choose $t_p$ as the outcome*

*4. else choose $\bar{t}_p$.*

**Lemma 5.2** *Mechanism 5.1 truthfully implements $f$ in dominant strategies in polynomial time for polynomial time bounded agents.*

*Proof.* Mechanism 5.1 is polynomial time since, to compute the outcome, it need only compare the number of clauses in $\theta$ that are satisfied by two fixed truth assignments. Truth telling is certainly a polynomial time strategy so it suffices to show that truth telling constitutes a dominant strategy equilibrium.

Let $\theta_i$ be the clause representing agent $i$'s true preference relation.

Case 1: $t_p$ satisfies $\theta_i$ and $\bar{t}_p$ doesn't.

Agent $i$ cannot change the number of clauses declared by the other agents that are satisfied by $t_p$ or $\bar{t}_p$. Therefore, by lying, agent $i$ can decrease (or leave unchanged) the number of clauses satisfied by $t_p$ and increase (or leave

unchanged) the number of clauses satisfied by $\bar{t}_p$. Thus, if lying has any effect at all, it is to change the outcome from $t_p$ to $\bar{t}_p$, which is worse for agent $i$.

Case 2: $\bar{t}_p$ satisfies $\theta_i$ and $t_p$ doesn't. A symmetric argument applies to this case.

Case 3: Both $t_p$ and $\bar{t}_p$ satisfy $\theta_i$. In this case, agent $i$ does not care which of the two truth assignments is chosen so lying cannot be beneficial or harmful.

Since lying is never beneficial, truth telling is a dominant strategy. ∎

**Theorem 5.1** $\frac{1}{2}$-MAXSAT *is strongly implementable in dominant strategies in polynomial time for polynomial time bounded agents.*

*Proof.* In the proof of Lemma 5.2, Case 3 is the only case in which lying can affect the outcome and not be harmful to the agent. Therefore, if the agents all play dominant strategies then the only agents that lie are indifferent to the outcome. We claim that if any group of agents that are indifferent between $t_p$ and $\bar{t}_p$ affect the outcome by lying, then the new outcome is still in $\frac{1}{2}$-MAXSAT.

Let $p$ be the problem instance and $\theta$ be the true preference profile. Suppose that a number of agents who are indifferent lie about their preference. Let $\theta'$ be the declared preference profile. Assume without loss of generality that the mechanism outputs $\bar{t}_p$ when the agents declare the true preference profile $\theta$, and $t_p$ when the agents declare a preference profile $\theta'$ which may include a number of lies. We know from Lemma 5.1 that, for any preference profile, at least one of $t_p$ or $\bar{t}_p$ satisfies at least half of the clauses. Therefore, $t_p$ must satisfy at least half of the clauses in $\theta'$. Let $b$ be the number agents that strictly prefer $t_p$ to $\bar{t}_p$ and let $d$ be the number of agents who are indifferent between $t_p$ and $\bar{t}_p$. Since only indifferent agents lie, the number of clauses satisfied by $t_p$ in $\theta'$ and in $\theta$ is $b + d$. Since the mechanism outputs $t_p$ when the message profile is $\theta'$, $b + d$ must be at least $I/2$ where $I$ is the total number or agents. But this implies $t_p \in \frac{1}{2}$-MAXSAT$(p, \theta)$. Therefore, every dominant strategy equilibrium outcome is in $\frac{1}{2}$-MAXSAT$(p, \theta)$ for every $p$ and $\theta$. ∎

## 6 Nash Implementation

We now consider mechanism design in environments with complete information using Nash equilibrium. There are two properties of social choice rules that are extremely important for Nash implementation – no veto power and monotonicity.

**Definition 6.1** *A social choice rule $F$ satisfies* **no veto power** *if, when all but at most one of the agents ranks an outcome $t$ as their weakly most preferred choice under a problem instance $p$ and preference profile $\theta$, $t \in F(p, \theta)$.*

**Definition 6.2** *For each agent $i$, outcome $t$, and preference relation $\theta_i$, define agent $i$'s* **lower contour set** *for $t$ and $\theta_i$ to be the set:*

$$L_i(t, \theta_i) = \{t' : t \text{ is preferred to } t' \text{under preference relation } \theta_i\}$$

*A social choice rule $F$ is said to be* **monotonic** *if, for all problem instances $p$, all preference profiles $\theta$ and $\theta'$, and all outcomes $t$ such that*

*(1)* $t \in F(p, \theta)$ *and*
*(2)* $L_i(t, \theta_i) \subseteq L_i(t, \theta_i')$ *for all $i$,*

$t \in F(p, \theta')$.

Condition 1, in Definition 6.2, states that $t$ is socially desirable under $\theta$. Condition 2 states that no agent prefers an outcome to $t$ under $\theta'$ that he does not prefer to $t$ under $\theta$. Monotonicity says that when these two conditions are satisfied, $t$ must be socially desirable under $\theta'$ as well.

As the following result known as Maskin's Theorem (see [15]) shows, monotonicity is a necessary and almost sufficient condition for full Nash implementation.

**Proposition 6.1 (Maskin's Theorem)** *If a social choice rule is fully Nash implementable then it is monotonic. If there are at least three agents then a social choice rule that is monotonic and satisfies no veto power is fully Nash implementable.*

Monotonicity is also required for full Nash implementation in polynomial time for polynomial time bounded agents.

**Proposition 6.2** *If a social choice rule $F$ is fully Nash implementable in polynomial time for polynomial time bounded agents then it is monotonic.*

*Proof.* The result follows from Propositions 3.1 and 6.1. ∎

Given Maskin's theorem we can show that MAXSAT is not fully implementable.

**Proposition 6.3** *The social choice rule for* MAXSAT *is not monotonic.*

*Proof.* Consider the problem instance $p$ in which there are two agents and two variables $x_1$ and $x_2$. Let $\theta = (x_1, \bar{x}_1)$. The maximum number of simultaneously satisfiable clauses in $\theta$ is 1. Let $t = x_1 x_2$ which is in MAXSAT$(p, \theta)$. Let $\theta' = (x_2, \bar{x}_1)$. Since $t$ satisfies agent 1 when the preference profile is either $\theta$ or $\theta'$, agent 1 weakly prefers $t$ to any other outcome under both $\theta_1$ and $\theta'_1$. Therefore, $L_1(t, \theta_1) = L_1(t, \theta'_1)$. Since agent 2's preference is the same in either case, $L_2(t, \theta_2) = L_2(t, \theta'_2)$. However, $t \notin$ MAXSAT$(p, \theta')$ which implies that MAXSAT is not monotonic. ∎

The argument in the above proof can easily be generalized to instances in which the number of variables and agents is larger, so there is no hope for achieving monotonicity on a (non-trivial) restricted set of problem instances.

*Approximation Mechanism for Nash Implementation*

Corollary 4.1 and Proposition 6.3 present two different obstacles to implementing MAXSAT in polynomial time for polynomial time bounded agents. Corollary 4.1 says that polynomial time implementation is impossible because of the computational constraints. Proposition 6.3 says that full implementation cannot be achieved even without the computational constraints. Given these obstacles, we would like to find a constant $c$ such that $c$-MAXSAT is strongly implementable in polynomial time for polynomial time bounded agents. If we were to find a $c$ such that $c$-MAXSAT is monotonic and satisfies no-veto

power then, by Maskin's Theorem, we would know that $c$-MAXSAT is fully implementable. This, however, would not guarantee that $c$-MAXSAT is fully implementable by a polynomial time mechanism. In fact, Theorem 6.1 below shows that the standard mechanisms used to prove Maskin's theorem are not polynomial time when the social choice rule is $c$-MAXSAT.

There are several different mechanisms used to prove Maskin's theorem. (See [15,31,33,17].) For example, in [31], Repullo defines a mechanism that fully implements a monotonic social choice rule satisfying no veto power as follows. Each agent $i$ declares a preference profile $\theta^i$, a proposed outcome $x^i$, and a number $k^i$. Let $p$ be the problem instance and let $m = [(\theta^i, x^i, k^i)]_{i=1}^{I}$ be the message profile. Let $g$ be the outcome function.

**Mechanism 6.1 (Repullo's Mechanism)**

*(1) If every agent picks the same message $(\theta, x, k)$ and $x \in F(p, \theta)$, then $g(p, m) = x$.*

*(2) If every agent but agent $i$ picks the same message $(\theta, x, k)$ and $x \in F(p, \theta)$, then*

$$g(p, m) = \begin{cases} x^i \text{ if } x^i \in L(x, \theta_i), \\ x \quad \text{otherwise} \end{cases}$$

*(3) If neither Rule 1 nor Rule 2 applies, then set $g(p, m) = x^j$ where $j$ is the agent with the highest $k^j$ and ties are broken by choosing the least such $j$.*

The idea behind the mechanism is that if everyone declares the true preference profile $\theta$, the same socially desirable outcome $x$, and the same number $k$, Rule 1 will select $x$ as the outcome. Rule 2 ensures that no single agent can profitably deviate from this strategy profile, establishing $x$ as a Nash equilibrium outcome. Rule 3 has the effect that, if neither Rule 1 nor Rule 2 applies, then any agent can deviate from the strategy profile and change the outcome to anything he wants. This eliminates undesirable equilibrium outcomes. (See [31] for details.)

The only computationally non-trivial part of this mechanism is checking to see if $x \in F(p, \theta)$. This check, which also appears in the mechanisms used in [33] and [15] corresponds to the following decision problem when $F = c$-MAXSAT

for some constant $c$.

**Definition 6.3** *Let $c, 0 < c < 1$ be fixed. The* **$c$-MAXSAT Membership Problem** *is defined as follows: Given a set of Boolean variables $V = \{v_1, \ldots, v_n\}$, a set of clauses $\theta$ over $V$, and a truth assignment $t$, does $t$ satisfy at least $c$ times the maximum number of simultaneously satisfiable clauses in $\theta$?*

Lemma 6.1 shows that for fixed $c$ this problem is NP-hard, which implies that Repullo's mechanism with $F = c$-MAXSAT is not a polynomial time mechanism.

**Lemma 6.1** *Let $c, 0 < c < 1$ be such that there is a polynomial time approximation algorithm for MAXSAT that guarantees the number of satisfied clauses is at least $c$ times the maximum number of simultaneously satisfiable clauses. The $c$-MAXSAT Membership problem is* NP-hard.

*Proof.* Suppose we are given a set of clauses $\theta$ and we want to determine whether there is a truth assignment that satisfies all the clauses in $\theta$. Let $m(\theta)$ be the maximum number of clauses in $\theta$ that can be simultaneously satisfied. The set of clauses $\theta$ is satisfiable if and only if $m(\theta) = I$ where $I$ is the number of clauses in $\theta$.

Consider the algorithm in Figure 1 for finding $m(\theta)$. We need to show that this algorithm returns the correct value of $m(\theta)$. Any truth assignment for $V^{k-1}$ can be extended to satisfy $v_{n+k}$ without affecting the number of clauses it satisfied in $\theta^{k-1}$ since, for each $k$, $v_{n+k}$ does not appear in any clause in $\theta^{k-1}$. Therefore, $m(\theta^k) = m(\theta) + k$. Since $t$ does not satisfy $v_{n+k}$ for any $k \geq 1$, $t$ satisfies $b$ of the clauses in each of the $\theta^k$. The repetition is guaranteed to terminate by the time $k = \lceil \frac{1-c}{c} m(\theta) \rceil + 1$ since $t$ satisfies at most $m(\theta)$ clauses in $\theta^k$ and $m(\theta^k) = m(\theta) + \lceil \frac{1-c}{c} m(\theta) \rceil + 1 \geq \frac{m(\theta)}{c} + 1$ when $k = \lceil \frac{1-c}{c} m(\theta) \rceil + 1$. Since the algorithm begins with a truth assignment that satisfies $b \geq cm(\theta)$ clauses, it stops extending the set of clauses when $k$ is the smallest integer such that $b < c(m(\theta) + k)$. This implies $m(\theta) = \lfloor (1/c)b \rfloor - k + 1$.

To evaluate the asymptotic running time of the algorithm, first note that, given our assumptions, we know that each of the steps of this algorithm other

**Compute max number satisfiable**

1 Use the polynomial time approximation algorithm we assumed we have for MAXSAT to choose a truth assignment $t$ that satisfies at least $cm(\theta)$ clauses.
2 Let $b$ be the number of clauses in $\theta$ that $t$ satisfies.
3 Let $k = 0$.
4 **repeat**
5 Increment $k$.
6 Extend the set of variables to the set $V^k$ by adding variable $v_{n+k}$.
7 Extend the set of clauses to $\theta^k$ by adding a clause with a single literal $v_{n+k}$.
8 Extend $t$ to $V^k$ by setting $t(v_{n+k})$ to False.
9 **until** $t$ extended to $V^k$ satisfies less than $cm(\theta^k)$ clauses.
10 **return** $\lfloor (1/c)b \rfloor - k + 1$.

Fig. 1. An algorithm for computing $m(\theta)$ using an algorithm to solve the $c$-MAXSAT Membership Problem in Step 9.

than checking the condition in Step 9 takes polynomial time. Furthermore, the maximum number of iterations is $\lceil \frac{1-c}{c} I \rceil + 1$, so since $c$ is fixed, there are a polynomial number of iterations. Hence, if Step 9 takes polynomial time, the entire algorithm is polynomial time. Since Step 9 is equivalent to checking whether $t \in c\text{-MAXSAT}(p, \theta^k)$, the entire algorithm is polynomial time if and only if the $c$-MAXSAT Membership problem can be solved in polynomial time.

We can use this algorithm to decide SAT by comparing the output to the number of clauses in $\theta$. The Boolean formula $\theta$ is satisfiable if and only if the value for $m(\theta)$ returned in Step 10 is $I$. Therefore, given an algorithm that solves the $c$-MAXSAT Membership problem in polynomial time, we can create an algorithm that solves SAT in polynomial time algorithm. Hence, the $c$-MAXSAT Membership problem is NP-hard. ∎

**Theorem 6.1** *Let $c, 0 < c < 1$, be such that there is a polynomial time approximation algorithm for $c$-MAXSAT. When $F = c$-MAXSAT, Repullo's mechanism is not a polynomial time mechanism.*

*Proof.* For the mechanism to check whether $x \in F(\theta)$ in Rules 1 and 2, it requires a solution to the $c$-MAXSAT Membership problem. Lemma 6.1 shows this cannot be done in polynomial time. ∎

For Repullo's mechanism to implement $c$-MAXSAT, for every problem in-

stance $p$ and preference profile $\theta$, at least one of the agents must choose an outcome that is in $c$-MAXSAT$(p, \theta)$. Theorem 6.1 implies that if $c$ is such that Repullo's mechanism with $F = c$-MAXSAT is polynomial time then the agents cannot find an alternative in $c$-MAXSAT in polynomial time. Therefore, either the mechanism is not polynomial time or there is no polynomial time equilibrium strategy profile.

However, as discussed in Section 2, we are willing to settle for strong implementation as opposed to full implementation of $c$-MAXSAT. Thus, it is sufficient for our purposes to find a social choice rule $F$ such that:

(1) $F$ is monotonic and satisfies no veto power
(2) $F(p, \theta) \subseteq c$-MAXSAT$(p, \theta)$ for all $p$ and $\theta$
(3) checking membership in $F$ is easy.

We have seen that $\frac{1}{2}$-MAXSAT is strongly implementable in dominant strategies using Mechanism 5.1. However, Mechanism 5.1 is not a particularly attractive mechanism since, for a problem instance $p$, the outcome is always either $t_p$ or $\bar{t}_p$ for a fixed $t_p$. An unfortunate choice of $t_p$ eliminates many outcomes that would be more desirable from a social viewpoint. For example, consider a problem instance with four agents and two variables $x_1$ and $x_2$. Let $t_p = x_1 x_2$, and the true preference profile $\theta = (x_1, x_1, \bar{x}_2, \bar{x}_2)$. The number of clauses in $\theta$ that are satisfied by $t_p$ and $\bar{t}_p$ is 2. However, it is possible to satisfy all of the agents by choosing $t' = x_1 \bar{x}_2$ as the outcome. It would be better to implement a social choice rule that included any truth assignment that satisfied as least as many clauses as both $t_p$ and $\bar{t}_p$. It could be argued that full Nash implementation of this social choice rule would be better than strong dominant strategy implementation using Mechanism 5.1. Unfortunately, this social choice rule is not fully Nash implementable (see [26]). Lemma 6.2 defines an alternative social choice rule that is fully Nash implementable and does not eliminate optimal outcomes from consideration.

**Lemma 6.2** *Let $F(p, \theta) = \{t : t$ satisfies at least half the clauses in $\theta\}$. If the set of problem instances is restricted to those in which there are at least three agents, $F$ is fully Nash implementable using Repullo's mechanism.*

*Proof.* $F$ satisfies no veto power since, if all but one agent ranks an out-

come $t$ as their (weakly) most preferred outcome, $t$ satisfies at least half of the agents.[3] Let $p$, $\theta$, $\theta'$ and $t$ be such that $t \in F(p,\theta)$ and $L_i(t,\theta_i) \subseteq L_i(t,\theta'_i)$ for all $i$. Suppose $t$ satisfies $\theta_j$ for some agent $j$. Then $L_j(t,\theta_j)$ contains every possible outcome and, since $L_j(t,\theta_j) \subseteq L_j(t,\theta'_j)$, $L_j(t,\theta'_j)$ contains every possible outcome. Since every clause has a satisfying truth assignment, this implies that $t$ satisfies $\theta'_j$. Since this is true for each $j$ such that $t$ satisfies $\theta_j$ and since $t \in F(p,\theta)$, $t$ satisfies at least half of the clauses in $\theta'$. This implies $t \in F(p,\theta')$. Therefore, $F$ is monotonic. Since $F$ is monotonic and satisfies no veto power, Repullo's mechanism fully Nash implements $F$. ∎

**Theorem 6.2** *When restricted to problem instances $p$ in which there are at least three agents, the social choice rule $\frac{1}{2}$-MAXSAT is strongly Nash implementable in polynomial time for polynomial time bounded agents.*

*Proof.* Let $F$ be the the social choice rule defined in Lemma 6.2. It is easy to check whether a truth assignment satisfies at least half of the total number of clauses, so Repullo's mechanism is polynomial time computable. Let $\theta$ be the agents' true preference profile. Let $t$ be any truth assignment in $F(p,\theta)$. The strategy profile in which each agent sends the message $(\theta, t, 1)$ is a Nash equilibrium. (See [31].) Such a strategy can be computed in polynomial time since the agents could, for example, all use the complement algorithm to find a $t$ that satisfies at least half of the total number of clauses. Hence, $F$ is fully Nash implementable in polynomial time for polynomial time bounded agents. Since $F(p,\theta) \subseteq \frac{1}{2}$-MAXSAT$(p,\theta)$ for all $p$ and $\theta$, $\frac{1}{2}$-MAXSAT is strongly Nash implementable in polynomial time for polynomial time bounded agents. ∎

Since $F$ is implementable and includes every optimal outcome, it follows from Lemma 3.1 that every optimal outcome is a polynomial time equilibrium outcome. Thus, while strong dominant strategy implementation using Mechanism 5.1 can eliminate optimal outcomes from occurring, full Nash implementation of $F$ using Repullo's mechanism would not preclude the possibility of any optimal outcome. The problem we face is that of defining mechanisms so

[3] Note that we are assuming it is possible to satisfy any individual agent. If an agent can specify an empty clause and we take this as an indication that the agent is never satisfied, the result does not hold. In particular, $F$ does not satisfy no veto power and is not monotonic.

that the suboptimal equilibrium outcomes are not too bad. As we show in the next section, our ability to eliminate suboptimal outcomes is severely limited.

## 7 Upper Bounds on Approximability

Existing work in mechanism design shows that it is possible to implement a wider range of social choice rules in undominated Nash equilibrium [4] [29,10] and subgame perfect equilibrium [5] [20] than in Nash equilibrium. In [29], it is observed that for a social choice rule to be fully implementable in Nash, undominated Nash, or subgame perfect equilibrium, it must satisfy Property Q defined below. Property Q must also be satisfied for full implementation in dominant strategies. Therefore, by Proposition 3.1, Property Q must be satisfied for full implementation in polynomial time for polynomial time bounded agents for each of these four equilibrium concepts.

**Definition 7.1 ([29])** *A social choice rule satisfies* **Property Q** *if, for any problem instance $p$, whenever $\theta$, $\theta'$ and $t$ are such that $t \in F(p,\theta)$ and $t \notin F(p,\theta')$, there is an agent $i$ such that $\theta_i \neq \theta'_i$ and agent $i$ is not completely indifferent under $\theta'_i$.*

Property Q says that, if an outcome goes from being socially desirably under a one preference profile to socially undesirable under a new preference profile, at least one of the agents whose preferences have changed is not completely indifferent under the new preference profile.

The following lemma shows that the only approximate social choice rules for MAXSAT that can be fully implemented are those that guarantee the number of clauses satisfied is a constant times the total number of clauses rather than a constant times the maximum number of simultaneously satisfiable clauses.

**Lemma 7.1** *Let $F$ be a social choice rule such that, for some constant $c, 0 < c \leq 1$, $F(p,\theta) \subseteq c$-MAXSAT$(p,\theta)$ for all $p$ and $\theta$. Suppose, for some problem*

[4] An undominated Nash equilibrium is a Nash equilibrium in which no agent plays a weakly dominated strategy.

[5] Subgame perfect equilibrium is defined for games that are played in stages. A strategy profile is a subgame perfect equilibrium if it is a Nash equilibrium in every subgame.

*instance $p$ and preference profile $\theta$, there exists a truth assignment $t \in F(p, \theta)$ that satisfies strictly less than $cI$ clauses where $I$ is the number of agents. Then $F$ does not satisfy Property Q.*

*Proof.* Let $F$, $p$, $\theta$, $c$, and $t$ be as defined above. For each $i$ such that $t$ satisfies $\theta_i$, let $l_i$ be any literal in $\theta_i$ such that $t(l_i) = \text{True}$. Create $\theta'$ from $\theta$ by adding $\bar{l}_i$ to every $\theta_i$ that $t$ satisfies. For each $i$, either $\theta'_i = \theta_i$ or agent $i$ is completely indifferent under $\theta'_i$. Hence, for $F$ to satisfy Property Q, it must be the case that $t \in F(p, \theta')$.

Let $N^*(\theta')$ be the maximum number of simultaneously satisfiable clauses in $\theta'$. Since $\bar{t}$ must satisfy every clause in $\theta'$ that $t$ does not satisfy and since $\bar{t}$ must satisfy $\theta'_i$ if $t$ satisfies $\theta_i$, $N^*(\theta') = I$. Since the clauses in $\theta$ that $t$ doesn't satisfy also appear in $\theta'$, the number of clauses in $\theta'$ that $t$ satisfies is strictly less than $cI = cN^*(\theta')$. Therefore, $t \notin c\text{-MAXSAT}(p, \theta')$ and, since $c\text{-MAXSAT}(p, \theta') \supseteq F(p, \theta')$, $t \notin F(p, \theta')$. Thus, $F$ does not satisfy Property Q. ∎

The following theorem shows that it is impossible to strongly implement $c$-MAXSAT if $c > 1/2$:

**Theorem 7.1** *Let $c$ be a constant such that $1/2 < c \leq 1$. Then $c$-MAXSAT cannot be strongly implemented in dominant strategy, Nash, undominated Nash or subgame perfect equilibrium.*

*Proof.* Let $\Gamma$ be any mechanism that strongly implements $c$-MAXSAT. Let $E(p, \theta)$ be the set of equilibrium outcomes of $\Gamma$ for problem instance $p$ and preference profile $\theta$. By definition, $\Gamma$ fully implements $E$. Since $\Gamma$ strongly implements $c$-MAXSAT, $E(p, \theta) \subseteq c\text{-MAXSAT}(p, \theta)$ for all $p$ and $\theta$.

Let $p$ be a problem instance such that the number of agents $I$ is even. Let $\theta'_i = x_1$ for $1 \leq i \leq I/2$. Let $\theta'_i = \bar{x}_1$ for $I/2 + 1 \leq i \leq I$. Since the maximum number of simultaneously satisfiable clauses is $I/2$, no truth assignment can satisfy more than $I/2$ clauses. Therefore, any $t \in E(p, \theta')$ satisfies less than $cI$ clauses in $\theta'$. According to Lemma 7.1, this implies that $E$ does not satisfy Property Q. But then $E$ is not strongly implementable which is a contradiction since $\Gamma$ fully implements $E$. ∎

By Proposition 3.1, Theorem 7.1 also holds for strong implementation in polynomial time for polynomial time bounded agents. Our inability to strongly implement $c$-MAXSAT for $c > 1/2$ is perhaps not surprising since Theorem 7.1 also implies that the exact social choice rule (take $c = 1$) is not strongly implementable even when there are no computational restrictions. In other words, we are attempting to use approxmiation not only to overcome the computational constraints of the problem but also to overcome the game theoretic constraints.

## 8 Conclusion

Using a multiagent version of MAXSAT, we have investigated the difficulties that arise in applying classic results from the mechanism design literature to computationally complex optimization problems. The following table summarizes the results presented in this paper.

Table 1
A summary of results on the implementability of MAXSAT and $c$-MAXSAT. [6]

| Rule | Dominant Strategy | Nash |
|---|---|---|
| MAXSAT | Truthfully<br>Not Truthfully in Poly Time<br>Not Strongly | Not Strongly |
| $\frac{1}{2}$-MAXSAT | Truthfully in Poly Time<br>Strongly in Poly Time | Strongly in Poly Time |
| $(> \frac{1}{2})$-MAXSAT | Not Strongly | Not Strongly |

We have demonstrated that, despite the impossibility results regarding dominant strategy implementation, it is possible to implement an approximate social choice rule for multiagent MAXSAT in dominant strategy equilibrium. We showed that $\frac{1}{2}$-MAXSAT is the best approximate social choice rule for

multiagent MAXSAT that can be strongly implemented in dominant strategy, Nash, undominated Nash or subgame perfect equilibrium in spite of the fact that there are many approximation algorithms for the non-multiagent version of this problem that achieve better lower bounds than 1/2. Several authors [39,8,1] provide algorithms for the non-multiagent version of MAXSAT that achieve lower bounds $\geq \frac{3}{4}$. Our results, therefore, indicate that it can be much more difficult to design good approximation mechanisms than to design good approximation algorithms. Future work should consider multiagent versions of other computational problems and should consider environments with incomplete information.

Our results suggest the following general approach to designing mechanisms for computational problems in non-quasilinear environments.

(1) Determine whether the Gibbard-Sattherthwaite Theorem applies. If it does, then dominant strategy implementation is not an option. If it doesn't apply and the agents are restricted to polynomial time, look for a direct revelation mechanism to truthfully implement an approximate social choice rule. If the agents are not restricted then non-revelation mechanisms should be considered.
(2) If dominant strategy implementation is not possible then find an approximate social choice rule $F$ such that:
    (a) $F$ satisfies Property Q.
    (b) $F$ is monotonic.
    (c) $F$ satisfies the no-veto property.
    (d) The problem of checking membership in $F$ is easy.

    If such an approximate social choice rule can be found then use Repullo's mechanism. (A similar mechanism from [33] may be more appropriate in some cases.)

*Acknowledgments*

We would like to thank Larry Kranich for many helpful comments and suggestions.

## References


[1] T. Asano. Approximation algorithms for MAX SAT: Yannakakis vs. Goemans and Williamson. In *Proceedings of the 3rd Israel Symposium on Theory and Computing Systems*, pages 24–37, Ramat Gan, Israel, 1997.

[2] R. Battiti. Approximation algorithms and heuristics for MAX-SAT. In D.-Z. Zhu and P.M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 1, pages 77–148. Kluwer Academic Publishers, Norwell, MA, 1998.

[3] E. Ben-Porath. Repeated games with finite automata. *Journal of Economic Theory*, 59:17–32, 1993.

[4] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158, New York, NY, 1971.

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[6] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[7] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41:587–602, 1973.

[8] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiabilty problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[9] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[10] M. Jackson, T. Palfrey, and S. Srivastava. Undominated Nash implementation in bounded mechanisms. *Games and Economic Behavior*, 6:474–501, 1994.

[11] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and Systems Sciences*, 9:256–278, 1974.

[12] S. Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, Cambridge, MA, 2001.

[13] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[14] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, New York, NY, 1995.

[15] E. Maskin. The theory of implementation in Nash equilibrium: a survey. In L. Hurwicz, D. Schmeidler, and H. Sonnenschein, editors, *Social goals and social organization: Essays in memory of Elisha Pazner*, pages 173–204. Cambridge University Press, New York, NY, 1985.

[16] E. Maskin and T. Sjöström. Implementation theory. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*. North-Holland, Amsterdam, 2002.

[17] R. McKelvey. Game forms for Nash implementation of general social choice correspondences. *Social Choice and Welfare*, 6:139–156, 1989.

[18] D. Monderer, M. Tennenholtz, and H. Varian. Special issue: Economics and artificial intelligence. *Games and Economic Behavior*, 35:1–5, 2001.

[19] J. Moore. Implementation, contracts and renegotiation in environments with complete information. In J.-J. Laffont, editor, *Advances in economic theory: Sixth World Congress*, pages 182–282. Cambridge University Press, Cambridge, UK, 1992.

[20] J. Moore and R. Repullo. Subgame perfect implementation. *Econometrica*, 56:1191–1220, 1988.

[21] A. Neyman. Cooperation, repetition and automata. In S. Hart and A. Mas-Colell, editors, *Cooperation: Game Theoretic Approaches*, volume NATO ASI–Series F, Vol. 155, pages 233–255. Springer Verlag, New York, NY, 1997.

[22] A. Neyman and D. Okada. Two person repeated games with finite automata. *Int. J. Game Theory*, 29(3):309–325, 2000.

[23] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 129–140, Atlanta, GA, 1999.

[24] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the Second ACM Conference on Electronic Commerce*, pages 242–252, October 2000.

[25] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[26] T. C. O'Connell. *Bounded Rationality in Repeated Games and Mechanism Design for Agents in Computational Settings.* PhD thesis, Department of Computer Science, University at Albany, SUNY, Albany, NY 12222, 2000.

[27] T. C. O'Connell and R. E. Stearns. Polynomial time mechanism design for collective decision making. In S. Parsons, P. Gmytrasiewicz, and M. J. Wooldridge, editors, *Game theory and decision theory in agent-based systems.* Kluwer, Boston, MA, 2002.

[28] T. C. O'Connell and R. E. Stearns. On finite strategy sets for finitely repeated zero sum games. *Games Econ. Behavior (to appear)*, 43:107–136, 2003.

[29] T. Palfrey and S. Srivastava. Nash implementation using undominated strategies. *Econometrica*, 59:479–501, 1991.

[30] C. Papadimitriou. On games with a bounded number of states. *Games and Economic Behavior*, 4:122–131, 1992.

[31] R. Repullo. A simple proof of Maskin's theorem on Nash implementation. *Social Choice and Welfare*, 4:39–41, 1987.

[32] J. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, Cambridge, MA, 1994.

[33] T. Saijo. Strategy space reductions in Maskin's theorem: Sufficient conditions for Nash implementation. *Econometrica*, 56(3):693–700, 1988.

[34] T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Mutiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press, Cambridge, MA, 1999.

[35] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[36] R. E. Stearns. Memory bounded game playing automata. Technical Report 547, Institute for Mathematical Studies in the Social Sciences, Stanford University, 1989.

[37] M. Tenneholtz, N. Kfir-Dahav, and D. Monderer. Mechanism design for resource bounded agents. In *Proceedings of the Fourth International Conference on Multiagent Systems*, pages 309–316, Boston, MA, 2000.

[38] V. V. Vazirani. *Approximation Algorithms*. Springer Verlag, New York, NY, 2001.

[39] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

## A P, NP and Approximation Algorithms

Viewed abstractly, programs take finite length strings as input and produce outputs. The input is thought of as a question and the output the answer. A "computational problem" can be thought of as a mapping of questions into answers. A program is said to **solve** a computational problem if it gives a correct answer to each question. A program which solves a problem is referred to as an **algorithm** for the problem and particular input sequences are referred to as **problem instances**. The complexity of an algorithm is measured by a *time function*. Time functions bound the "worst case" time, namely the largest number of steps taken by any input of a given length. The "number of steps used" is referred to as the **computation time**.

Polynomial time is defined as follows:

**Definition A.1** *An algorithm is said to be* **polynomial time** *if its computation time (as a function of input length) is bounded by a polynomial function.*

It is unknown whether polynomial time algorithms exist for some problems. For example, no polynomial time algorithm has been found to determine whether an arbitrary CNF Boolean formula is satisfiable. This problem is known in the computer science literature as the Satisfiability Problem or simply SAT. Technically, SAT is not formally defined until one specifies the notation whereby lists of clauses are to be described and presented as input to a program for solving the problem. There are obviously many natural ways to do this and the actual number of input symbols needed to describe a particular set of clauses will vary somewhat with the notation. However, the effects of these variations on the time functions are too small to make a polynomial algorithm non-polynomial or vice versa. Thus it makes sense to ask the question "can SAT be solved in polynomial time?" without being specific about the notation.

SAT is an example of a "YES/NO problem", namely a problem where the answer for any problem instance is YES or NO. YES/NO problems are called **decision problems**. Other problems require an answer describing a maximum or a minimum. These problems are referred to as **optimization problems**. One such problem is a variation on SAT known as MAXSAT .

**Definition A.2** *The problem* **MAXSAT** *is the following: given a list of clauses, what is the maximum number of clauses that can be simultaneously satisfied by an assignment to the variables?*

MAXSAT is the problem upon which the results in this paper are based. It is not known whether MAXSAT can be solved in polynomial time. However, it is obvious that MAXSAT cannot be easier than SAT, because an answer to MAXSAT gives an immediate answer to SAT. If the maximum number of clauses that can be satisfied is equal to the number of clauses, the answer to SAT is YES; and otherwise it is NO. Thus any evidence that SAT is hard is also evidence that MAXSAT is hard.

If a set of clauses is satisfiable, there is an easy proof that the SAT answer to the problem instance is YES. The proof consists of a proposed assignment to the Boolean variables and verification that the assignment satisfies each clause. By "easy", we mean that the proof can be checked in time polynomial in the instance length. Be sure to notice the difference between checking a proof (easy) and finding a proof (believed to be hard).

The class of YES/NO problems with easy to check proofs is called NP. The NP stands for "nondeterministic polynomial" which means that, if you guess a proof (guessing is nondeterministic) you can verify the proof in polynomial time. A major question in computer science and mathematics is "can all problems in NP be solved in polynomial time?" Letting P be the set of YES/NO problems solvable in polynomial time, the question is expressed simply as "does P = NP?"

In [4], it was proved that, if SAT is in P, then P = NP. In other words, if there are any problems in NP that are not in P, then SAT is one of these problems. Since it is hard to believe that all problems in NP can be solved in polynomial time, it is hard to believe that SAT can be solved in polynomial time. The key idea in Cook's proof is that, given a solver for SAT, only a polynomial number of additional steps would be needed to solve any problem in NP. A problem with this property is called **NP-hard**. Because of Cook's result, NP-hardness can be proven by showing that only a polynomial number of additional steps are needed to solve SAT (or any other known NP-hard problem). MAXSAT is NP-hard because, given a solver for MAXSAT, the answer to SAT is obtained

simply by comparing the maximum number of satisfiable clauses with the number of clauses.

*Approximation Algorithms*

When faced with an NP-hard optimization problem, we cannot hope to find a polynomial time algorithm assuming P $\neq$ NP. However, it may be that we would be willing to settle for approximately optimal solutions. In that case, we would like to develop a polynomial time *approximation algorithm*. For example, in [11], Johnson provides two polynomial time approximation algorithms for MAXSAT. Johnson's first approximation algorithm for MAXSAT guarantees that at least 1/2 of the maximum number of simultaneously satisfiable clauses are satisfied while his second algorithm guarantees that the number of clauses satisfied will be within a factor of 2/3 of the optimal. Several authors [39,8,1] provide algorithms for MAXSAT that achieve lower bounds $\geq 3/4$. (See [2] for a survey of approximation algorithms for MAXSAT and [38] for an introduction to approximation algorithms.)

When faced with the problem of designing mechanisms that must solve computationally difficult problems, we will no doubt have to resort to mechanisms that find approximately optimal solutions. For example, in [25] an existing approximation algorithm for the task scheduling problem was used to create a truthful mechanism for their multiagent version of the problem. In this paper, we showed that creating a good approximation mechanism is not always an easy thing to do even when we have several good approximation algorithms for the problem at hand.