# Implementing the Maximum of Monotone Algorithms

**Liad Blumrosen**
Microsoft Research, Silicon Valley
1065 La Avenida
Mountain View, California 94043

**Abstract**

Running several sub-optimal algorithms and choosing the optimal one is a common procedure in computer science, most notably in the design of approximation algorithms. This paper deals with one significant flaw of this technique in environments where the inputs are provided by selfish agents: such protocols are not necessarily incentive compatible even when the underlying algorithms are. We characterize sufficient and necessary conditions for such best-outcome protocols to be incentive compatible in a general model for agents with one-dimensional private data. We show how our techniques apply in several settings.

## Introduction

Decision making under incomplete information have received much attention in computer science in recent years. In such systems, the decision maker does not have direct access to the input of the protocol, which is private information of a set of selfish agents. These agents may manipulate the protocol by misreporting this data if it is beneficial for them. In the presence of selfish agents, systems should therefore guarantee that the desired outcome is obtained in equilibrium. Using standard economic terminology, we then say that this outcome (e.g., maximizing the social welfare or the mechanism's profit) is *implementable*. In many settings, determining the optimal outcome is computationally hard, even without taking incentive considerations into account, and approximation algorithms or other ad-hoc heuristics should be developed. One fundamental technique for constructing such algorithms runs several algorithms and chooses the result that is achieved by the best algorithm. In the design of approximation algorithms, this technique is usually used for balancing the worst-case performance of several extreme scenarios. See, e.g., (Mua'lem & Nisan 2002; Dobzinski, Nisan, & Schapira 2005; Briest & Krysta 2006; Babaioff & Blumrosen 2006) for few recent papers that use this popular technique for approximating economic objectives. Another prominent use of this technique is in the design of randomized algorithms, where algorithms are run repeatedly for boosting their success probability, see, e.g., (Motwani & Raghavan 1995). In mechanism design, however, this method cannot be directly applied, since taking the maximum over a set of algorithms does not preserve incentive compatibility. Namely, even when the underlying algorithms obtain their results in a truthful dominant-strategy equilibrium, there is no guarantee that their maximum will remain truthful. The goal of this paper is to devise a general characterization of algorithms whose maximum can be implemented in dominant strategies.

When considering a single algorithm, a full characterization of algorithms that are dominant-strategy implementable is known. This characterization holds when the agents have one-dimensional private data (e.g., a real number), and claims that the *monotonicity* of the algorithm is a sufficient and necessary condition for incentive compatibility. Roughly speaking, a monotone algorithm chooses a "better" output for a certain agent after the type of this agent was raised (a formal definition will be given later). Given a monotone algorithm, the payment scheme that supports a dominant-strategy equilibrium is uniquely defined. The outcome of each algorithm is associated with some *social value*, and the maximum of several algorithms chooses the output of the algorithm with the highest social value. This paper studies requirements from algorithms that guarantee that their maximum can be implemented in dominant strategies. We study this question in a general model that was recently introduced by (Blumrosen & Feldman 2006). This general model does not assume that the agents divide the possible outcomes to "losing" and "winning" outcomes; instead, they can rank the possible outcomes in arbitrarily delicate way. Also, the model allows general objective functions and it is not limited to maximizing social welfare.

### Our contribution

We present two characterization results. The first is a necessary and sufficient condition for the implementability in dominant strategies of the maximum over a set of algorithms. Informally speaking, this characterization observes the algorithms as collections of functions that map the type of one of the agents to an alternative, fixing all other types. Each algorithm is defined by the points at which these functions switch between alternatives as the private type of the particular agent increases. It turns out that these are exactly the points that are vulnerable to manipulation when a maxi-

mum is taken over the algorithms. This discontinuity of the output of the algorithms requires us to use somewhat sophisticated notations for defining these points. We are then able to define the *crossing points* of every pair of algorithms – these are profiles of types where *different* algorithms maximize social value when the type of some agent is slightly increased or slightly decreased. Finally, we show that the maximum over the algorithms is dominant-strategy implementable if and only if the alternatives chosen near these crossing points change monotonically.

The above characterization describes how the decisions of two algorithms should interact to guarantee that their maximum is incentive compatible. We would also like to devise a safety condition for a single algorithm, saying that maximum over algorithms for which this condition is met must be dominant-strategy implementable. Our second characterization result therefore provides a general class of algorithms for which this safety property holds. This property says that at every point that is a *potential* crossing point with another algorithm, the algorithm should choose the alternatives in a "safe" manner.

Finally, we demonstrate our results in two auction settings. We first consider an auction for rectangular bundles on the plane (e.g., in real-estate auctions). For this setting, (Babaioff & Blumrosen 2006) presented a dominant-strategy incentive-compatible mechanism that achieves the best approximation ratio currently known for this algorithmic problem when the agents are *single minded* (i.e., each agent is only interested in one particular bundle). However, the fact that an algorithm is monotone for single-minded bidders does not imply that it will also be monotone for more general valuation classes. We use our general characterization to construct an incentive-compatible mechanism for this problem in environments where each bidder is interested in several bundles with different values. Note that we still consider a single-parameter model, hence all of these values are functions of the type of that agent, linear functions in our case.

Although our second characterization result seems to be quite restrictive in the general setting, it turns out to be useful for identifying "safety" properties for specific environments. We apply this result to combinatorial-auction settings, and show how it can easily derive the neat class of "bitonic" algorithms introduced by (Mua'lem & Nisan 2002).

## Related Work

Single-parameter domains in mechanism design have been recently extensively studied in computer science. Examples include work on single-minded combinatorial auctions (e.g., (Lehmann, O'Callaghan, & Shoham 2002; Babaioff, Lavi, & Pavlov 2005)), auctions for digital goods (see (Goldberg *et al.* 2006) and the references within) and scheduling (e.g., (Andelman, Azar, & Sorani. 2005)). The closest result in spirit to this paper is by (Mua'lem & Nisan 2002), who studied the monotonicity of the maximum of two algorithms for single-minded combinatorial auctions. A further discussion of similar operators was given by (Kao, Li, & Wang 2005). The focus on single-parameter domains is owing to the fact that the space of algorithms that are dominant-strategy implementable in multi-parameter domains is not well understood. The only general characterization which is known is of VCG mechanisms, which can only implement a particular objective function (maximizing social welfare) and, moreover, require finding an exact optimal solution (e.g., (Nisan & Ronen 2001)).

# The Model

In this section, we describe a general mechanism-design model for single-parameter agents and discuss dominant-strategy implementation.

## A General Mechanism-Design Model

Consider a set of alternatives $ALT = \{a_1, ..., a_m\}$ and a set of $n$ agents. Each agent has a privately known type $\theta_i \in [\underline{\theta}_i, \overline{\theta}_i]$ (where $\underline{\theta}_i, \overline{\theta}_i \in \mathbb{R}$, $\underline{\theta}_i < \overline{\theta}_i$), and a type-dependent valuation function $v_i(\theta_i, a)$ (continuous and differentiable in $\theta_i$) for each alternative $a \in ALT$.[1] In other words, agent $i$ with type $\theta_i$ is willing to pay an amount of $v_i(\theta_i, a)$ for alternative $a$ to be chosen. We denote a profile of types by $\theta = (\theta_1, ..., \theta_n)$ and the set of all possible type profiles by $\Theta = \times_{i=1}^{n}[\underline{\theta}_i, \overline{\theta}_i]$. We use the notation $\theta_{-i}$ to denote the type profile excluding $\theta_i$, and we will also use the notation $\theta = (\theta_i, \theta_{-i})$. The utility of each agent is quasi-linear.

The social value that the social planner gains from each alternative depends on the types of the agents and is given by a *social-value function* $g : \Theta \times ALT \to \mathbb{R}$. Typically, the social planner aims to choose alternatives that maximize social value. We assume that for every alternative $a$, the function $g(\theta, a)$ is continuous with respect to all $\theta_i$. Note that the *social value* is a general concept, and it does not necessarily refer to the *social welfare* (which equals the sum of the valuations of the agents).

**Definition 1.** *A mechanism is a pair of functions where:*

- $A : \Theta \to ALT$ *is the algorithm (the allocation rule).*
- $p : \Theta \to \mathbb{R}^n$ *is the payment scheme (i.e.,* $p_i(\theta)$ *is the payment to the ith agent given a declared type profile* $\theta$*).*

## The MAX Operator for Algorithms

Given the above definitions, we can formally define the maximum over a set of algorithms.

**Definition 2.** *Given a set of algorithms* $A_1, ..., A_k$*, let* $max\{A_1, ..., A_k\}$ *denote an algorithm that chooses alternatives that maximize social value, that is, for every* $\theta \in \Theta$

$$\max\{A_1, ..., A_k\}(\theta) = A_j(\theta)$$

*where* $j \in argmax_{i \in \{1,...,k\}} g(A_i(\theta), \theta)$.

One subtle issue is the tie-breaking rule used by the MAX operator. Our results hold for any consistent (deterministic) tie breaking rule. Given a tie breaking rule, the above MAX operator is uniquely defined. To simplify the presentation of the rest of this paper, we will assume that the social-value

[1] For example, in a standard auction setting, $\theta_1$ is the value of Agent 1 for the item, and $v_1(\theta_1, "1\ wins") = \theta_1$ and $v_1(\theta_1, "2\ wins") = 0$.

curves $g(\cdot, \theta_{-i})$ do not coincide in any interval, except possibly crossing at a finite number of points. This assumption can be easily relaxed by simply taking the tie-breaking rule into account when comparing the social value; for example, if we write that $g(A_1(\theta), \theta) > g(A_2(\theta), \theta)$ it means that $A_1$ achieves higher welfare, or in case of a tie, the tie-breaking rule breaks the tie in favor of $A_1$.

### Dominant-Strategy Implementation

**Definition 3.** *An algorithm $A$ is dominant-strategy implementable (or just implementable) if there exists a payment scheme $p$ such that reporting the true type is the best declaration for each agent, regardless of the declarations of the other agents. That is, for every agent $i$ with a true type $\theta_i$ and for every declaration $\theta'_i$, and for every profile of types $\theta_{-i}$ of the other agents we have that:*

$$v_i(\theta, A(\theta)) - p_i(\theta) \geq v_i(\theta, A(\theta'_i, \theta_{-i})) - p_i(\theta'_i, \theta_{-i})$$

For characterizing the set of implementable algorithms, we will require a structural assumption on the preferences of the agents. This assumption, known as the *single-crossing condition* in the literature (or the Spence-Mirrlees condition), claims that unless two alternatives are "equivalent", the value of the agent for one alternative grows faster then its value for the other alternative. One consequence of this assumption is that the curves representing these values as a function of the type of a particular agent will cross at most once. This assumption is made, implicitly or explicitly, in almost every paper on mechanism design in single-parameter environments. The following definition is a variant of several properties suggested by earlier papers (see, e.g., the discussion in (Edlin & Shannon 1998)), designed for environments with a discrete space of alternatives and a continuum of types.

**Definition 4.** *A valuation function $v_i : [\underline{\theta}_i, \overline{\theta}_i] \times ALT \to \mathbb{R}$ is* single crossing *if there is an order $\succ_i$ on the alternatives, such that for any two alternatives $a_j \succ_i a_l$ we have that for every $\theta_i$,*

$$\frac{\partial v_i(\theta_i, a_j)}{\partial \theta_i} > \frac{\partial v_i(\theta_i, a_l)}{\partial \theta_i}$$

*and if neither $a_l \succ_i a_j$ nor $a_j \succ_i a_l$ (denoted by $a_j \sim a_l$) then $v_i(\cdot, a_j) \equiv v_i(\cdot, a_l)$ (i.e., the functions are identical).*

**Example 1.** ***(Single-minded combinatorial auctions)*** *Consider an auction for 3 items $x, y$ and $z$ among 5 agents. Each agent is only interested in a single bundle. Specifically, assume that agent 1 is willing to pay 3 for the bundle $xy$, denoted by $v_1(xy) = 3$. Similarly, $v_2(xy) = 6$, $v_3(yz) = 5$ and $v_4(yz) = 7$. We also know that Agent 5 is interested in the bundle $z$. There are 8 alternatives (here: feasible allocations of the items) in this model: $a_1$="1 wins", $a_2$="3 wins", $a_3$="2 wins", $a_4$="4 wins", $a_5$ = "5 wins", $a_6$ = "1 and 5 win", $a_7$="2 and 5 win", and also $a_0$="no one wins". For every alternative in which agent 5 loses ($a_1 - a_4$), the value of agent 5 for this outcome does not depend on $\theta_5$. When agent 5 wins, the welfare is linear in his type, $v_5(a_5, \theta_i) = \theta_i$. Clearly, these functions are single crossing (the slope of the winning alternatives is higher than the slope of the losing alternatives at every point). Figure 1 describes the* social value *(or the social welfare in this example) as a function of $\theta_5$ when all other types are fixed to their true values. Note that the when agent 5 loses, the social welfare is also independent of his type, and therefore such alternatives are illustrated as horizontal lines; for winning alternatives, the welfare increases linearly with the agent's type, and therefore it is depicted as 45 degree lines.*

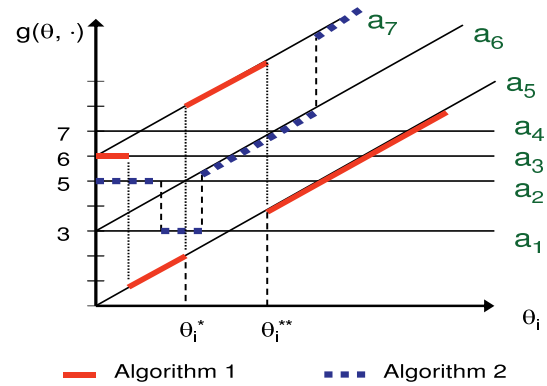


Figure 1: This figure describes the decisions made by two algorithms for the combinatorial-auction instance in Example 1. The $45°$ curves are equivalent with respect to agent $i$, and that alternatives $a_1 - a_4$ are equivalent (but inferior) for this agent. However, $\theta_i^{**}$ is a non-monotone *potential* crossing point for Algorithm 1 since the maximum of Algorithm 1 and an algorithm that chooses $a_4$ in the right neighborhood of $\theta_i^{**}$ would be non monotone.

It turns out that for dominant-strategy implementability, the decisions of the algorithms should be aligned with the single-crossing preferences.

**Definition 5.** *A mechanism is* monotone *if when agent $i$ raises his reported type, and fixing the types of the other agents, the mechanism never chooses an inferior alternative for $i$. That is, for any $\theta_{-i}$ if $\theta'_i > \theta_i$ then $A(\theta'_i, \theta_{-i}) \succeq_i A(\theta_i, \theta_{-i})$. (We denote by $a \succeq_i b$ cases where either $a \succ_i b$ or $a \sim b$.)*

For single-crossing preferences, monotonicity is a necessary and sufficient condition for dominant-strategy implementation. We refer the reader to, e.g., (Hermalin 2005; Mookherjee & Reichelstein 1992; Segal 2003) for the formal details.

**Proposition 1.** *Assume that the valuation functions $v_i(\theta_i, A)$ are single crossing for every agent $i$. An algorithm $A$ is dominant-strategy implementable if and only if $A$ is monotone.*

## Implementing the Maximum

We will first present a technical notation for the neighborhood of a profile of types. Such a neighborhood is a set of similar profiles in which the type of one of the agents is slightly changed.

**Definition 6.** *Consider an agent $i$ and a type profile $\theta \in \Theta$.*

- *The set of type profiles $L \subset \Theta$ is called a* left $i$-neighborhood of $\theta$, *if there exists $\delta > 0$ such that*

$$L = \{(\theta'_i, \theta_{-i}) \mid \theta_i - \delta < \theta'_i < \theta_i\}$$

- *The set of type profiles $R \subset \Theta$ is called a* right $i$-neighborhood of $\theta$, *if there exists $\delta > 0$ such that*

$$R = \{(\theta_i'', \theta_{-i}) \mid \theta_i < \theta_i'' < \theta_i + \delta\}$$

We will now formally define crossing points between algorithms. As mentioned, the decisions made by such algorithms will typically be discontinuous. The following definition thus captured crossing points of continuous and discontinuous functions. We will assume, for simplicity, that every algorithm admits a countable number of points in which it shifts between alternatives (for every fixed $\theta_{-i}$), and also that the algorithm do not have such shifts on singleton points.[2]

**Definition 7.** *Two algorithms $A$ and $B$* cross at $\theta \in \Theta$ with respect to agent $i$ *if there exists a left $i$-neighborhood $L$ and a right $i$-neighborhood $R$ to $\theta$ such that one of the following holds for every $\theta' \in L$ and every $\theta'' \in R$:*

$$g(A(\theta'), \theta') > g(B(\theta'), \theta') \quad \textit{and} \tag{1}$$
$$g(A(\theta''), \theta'') < g(B(\theta''), \theta'') \tag{2}$$

*or*

$$g(A(\theta'), \theta') < g(B(\theta'), \theta') \quad \textit{and} \tag{3}$$
$$g(A(\theta''), \theta'') > g(B(\theta''), \theta'') \tag{4}$$

For example, consider the combinatorial auction setting described in Example 1 and in Figure 1. The thick curve and the dashed thick curve in the figure describe the choices made by two algorithms denoted, respectively, as Algorithm 1 and Algorithm 2. The algorithms are crossing at, for example, $(\theta_i^*, \theta_{-i})$ and $(\theta_i^{**}, \theta_{-i})$ (where $i = 5$). For $\theta_i^*$, Algorithm 1 gains higher social value on its right neighborhood, and Algorithm 2 does so on the left neighborhood.

By definition, in the left neighborhood of the crossing point, one algorithm obtains a higher social value choosing some alternative $a$, and in the right crossing point this is done by the other algorithm choosing an alternative $b$. If the alternative $b$ is at least as good for agent $i$ as $a$ (according to the priority that was defined by the single-crossing structure, i.e., $b \succ_i a$) then this crossing point is $monotone$.

**Definition 8.** *Consider two algorithms $A$ and $B$ that are crossing with respect to agent $i$ at $\theta \in \Theta$. If there exist left and right $i$-neighborhoods with the same properties as in Definition 7 such that for every $\theta' \in L$ and $\theta'' \in R$, we have that $B(\theta'') \succeq_i A(\theta')$ (when Equations 1,2 hold) or $A(\theta'') \succeq_i B(\theta')$ (when Equations 3,4 hold) then $\theta$ is a* monotone crossing point of $A$ and $B$ with respect to agent $i$.

For example, in Figure 1, the algorithms are crossing at the point $(\theta_i^*, \theta_{-i})$. This crossing point is monotone, since the alternative chosen by the algorithm with the higher welfare in the left neighborhood ($a_1$) has smaller priority according to $\succ_i$ than the alternative chosen by the higher-value algorithm on the right neighborhood ($a_7$).

If all the crossing points are monotone, then two algorithms are *monotonically crossing*.

[2]Our results hold even without these assumptions, but this adds to the complexity of the presentation.

**Definition 9.** *Two algorithms $A, B$ are* monotonically crossing, *if for every agent $i$, all the crossing points of $A$ and $B$ with respect to agent $i$ are monotone.*

We can now present a necessary and sufficient condition for the implementability of $MAX\{A, B\}$:

**Theorem 1.** *Consider two monotone algorithms $A, B$. $\max\{A, B\}$ is implementable if and only if $A$ and $B$ are monotonically crossing.*

*Proof.* (sketch) Assume that $A$ and $B$ cross non monotonically. That is, there exists a type profile $\theta$ for which there is some point $\theta' = (\theta_i', \theta_{-i})$ in the left $i$-neighborhood of $\theta$, and a point $\theta'' = (\theta_i'', \theta_{-i})$ in the right $i$-neighborhood of $\theta$ such that (w.l.o.g.)

$$g(A(\theta'), \theta') > g(B(\theta'), \theta') \quad \text{and}$$
$$g(A(\theta''), \theta'') < g(B(\theta''), \theta'')$$

and also $A(\theta') \succ_i B(\theta'')$. Due to the definition of $\max\{A, B\}$, the alternative $A(\theta')$ will be chosen for the type profile $\theta'$, and $B(\theta'')$ will be chosen for $\theta''$. Since $\theta_i' < \theta_i''$ and $A(\theta') \succ_i B(\theta'')$, the algorithm $\max\{A, B\}$ is not monotone and therefore it is not implementable.

Conversely, assume that $A$ and $B$ are monotonically crossing. Now, let $\theta$ be a type profile for which $g(A(\theta), \theta) > g(B(\theta), \theta)$ and let $a = A(\theta)$. Denote $\theta'' = (\theta_i'', \theta_{-i})$, where $\theta_i'' = \sup\{x \mid \forall y \in [\theta_i, x],$
$g(A(y, \theta_{-i}), (y, \theta_{-i})) > g(B(y, \theta_{-i}), (y, \theta_{-i}))\}$.

We will show that $MAX(A, B)$ is monotone when $\theta_i$ increases. If $\theta_i'' = \overline{\theta}_i$, we are done. If there is a right $i$-neighborhood of $\theta''$ where both algorithms choose the same alternative, then this decision is monotone by the monotonicity of $A$ and $B$. Else, $\theta''$ is a crossing point between $A$ and $B$: it has a right $i$-neighborhood $R$ and for every type profile $\theta^*$ in $R$ we have $g(A(\theta^*), \theta^*) < g(B(\theta^*), \theta^*)$.[3] Denote the alternative chosen in this right neighborhood as $r$. Since the algorithms are monotonically crossing, $r$ has a higher priority with respect to agent $i$ than the alternative in the left neighborhood, and since $A$ is monotone we clearly also have that $r \succeq_i a$. Since the number of crossing points is countable, this argument inductively derives that the maximum of the algorithms is also monotone. □

**Corollary 1.** *Choosing the best outcome over every subset of monotone algorithms $A_1, ..., A_k$ is implementable if and only if every two algorithms are monotonically crossing.*

## Safe Algorithms for the MAX operator

Our next goal is to describe a property of algorithms such that the maximum of algorithms with such property always results in a monotone algorithm. We denote such algorithms as *MAX-safe* algorithms.

**Definition 10.** *Algorithm $A$ is* MAX-safe, *if for every set of algorithms $\{A_1, ..., A_k\}$, if $MAX\{A_1, ..., A_k\}$ is implementable then $MAX\{A_1, ..., A_k, A\}$ is also implementable.*

[3]Here we use the assumption that we made, for simplicity, that the algorithms switch between alternatives over intervals and not over singletons

Again, we will be interested in points that may potentially turn into crossing points between algorithms.

**Definition 11.** *A profile of types $\theta \in \Theta$ is a* potential crossing point *for algorithm $A$ with respect to agent $i$ if there exist a left $i$-neighborhood $L$, a right $i$-neighborhood $R$ for $\theta$, and two alternatives $l, r \in ALT$, such that for every two points $\theta' \in L, \theta'' \in R$ we have that either*

$$g(A(\theta'), \theta') < g(l, \theta') \text{ and } g(A(\theta''), \theta'') > g(r, \theta'') \quad (5)$$

*or*

$$g(A(\theta'), \theta') > g(l, \theta') \text{ and } g(A(\theta''), \theta'') < g(r, \theta'') \quad (6)$$

**Definition 12.** *Consider a potential crossing point $\theta$ for algorithm $A$ with respect to agent $i$. $\theta$ is* non-monotone*, if there exist left and right $i$-neighborhoods $L$ and $R$ with the same properties as in Definition 11, such that for every two type profiles $\theta' \in L$ and $\theta'' \in R$ we have that either $l \succ_i A(\theta'')$ (if Equation 5 holds) or $A(\theta') \succ_i r$ (if Equation 6 holds).*

**Definition 13.** *An algorithm $A$ has* monotone potential crossing points *if, for every agent $i$, $A$ does not possess non-monotone potential crossing points with respect to agent $i$.*

For example, Algorithm 1 in Figure 1 has a non-monotone potential crossing point with respect to agent $i$ at $\theta_i^{**}$: there is an alternative, $a_4$, that achieves a higher value on the right neighborhood, but the higher-priority alternative $a_7$ achieves the higher value on the left neighborhood. The next theorem shows that the property of having monotone potential crossing points actually characterizes MAX-safe algorithms; in our example, Algorithm 1 is not MAX-safe, and indeed, the maximum over Algorithm 1 and an algorithm that always chooses the alternative $a_4$ is non-monotone.

**Theorem 2.** *An algorithm is MAX-safe if and only if it is monotone and it has monotone potential crossing points.*

*Proof.* First, we show that if an algorithm $A_1$ admits non-monotone potential crossing points, then it is not MAX-safe. Assume that $\theta$ is a non-monotone potential crossing point for algorithm $A_1$ with respect to agent $i$. Then, there is a left $i$-neighborhood L, a right $i$-neighborhood $R$, and two alternatives $l \succ_i r$ such that for every $\theta' \in L$ and $\theta'' \in R$ we have that $g(A(\theta'), \theta') < g(l, \theta')$ and $g(A(\theta''), \theta'') > g(r, \theta'')$ and $l \succ_i A(\theta'')$. (The proof for the other type of non-monotone crossing points (Definition 8) is similar.) Consider now another algorithm $A_2$ that chooses the alternative $l$ for every point in $L$ and chooses $r$ for every point in $R$. By definition, the algorithm $MAX\{A_1, A_2\}$ will clearly choose the alternative $l$ at any point in $L$, and choose $A(\theta'')$ at every point in $R$. Since $l \succ_i A(\theta'')$, then $max\{A_1, A_2\}$ is not monotone, and thus non-implementable.

The other direction is an immediate corollary of Theorem 1: if an algorithm has monotone potential crossing points, then, by definition, it monotonically crosses any other algorithm, so their maximum is implementable. □

## Applications

This section presents applications of Theorems 1 and 2.

### Auctions for Rectangular Bundles

Consider the following problem: a seller wishes to sell properties on a 2-dimensional plane and agents have privately known values for different axis-parallel rectangles. The (NP-hard) objective of the seller is to allocate disjoint rectangles to agents such that the total sum of their values (the social welfare) is maximized. (Babaioff & Blumrosen 2006) showed how a variant of the "Shifting" algorithm by (Khanna, Muthukrishnan, & Paterson. 1998) forms a dominant-strategy incentive-compatible mechanism that guarantees an $O(\log R)$ approximation to the social welfare, where $R$ is an upper bound on the ratio of any two edges of rectangles. A central assumption made by Babaioff et al. is that the agents are single minded, that is, each agent may be interested in a single rectangle only, and her value for any bundle that does not contain this particular bundle is zero. Here we show that the Shifting mechanism is incentive compatible in a more general setting where each agent is interested in an unlimited number of bundles with different values. Since we remain in the single-parameter domain, all these values are linear functions of the type of agent $i$.

**Definition 14.** *An agent $i$ has an OR-linear valuation if there exist bundles $S_1, ..., S_{n_i}$ and coefficients $\alpha_1, ..., \alpha_{n_i}$ such that for any bundle $T$ the agent's value for receiving $T$ is $v_i(T) = max_I \sum_{j \in I} \alpha_j \theta_i$, where the maximum is taken over all the index sets $I$ such that for every $i \neq j \in I$, $S_j, S_i \subseteq T$ and $S_i \cap S_j = \emptyset$.*

Note that such valuations clearly generalize single-minded preferences (the agents are single minded when $n_i = 1$ and $\alpha_1 = 1$ for every agent). It is easy to see that such valuations are single crossing.

The Shifting mechanism actually runs several separate algorithms, and chooses the result that achieves the highest social welfare. Therefore this algorithm provides a good environment to demonstrate our techniques.

**Proposition 2.** *There exists a mechanism with a polynomial running time*[4] *that achieves an $O(\log R)$ approximation to the optimal welfare and is incentive compatible for agents with OR-linear valuations.*

The description of the shifting mechanism is involved and requires many details. Due to lack of space, we will only mention an outline of this algorithm and of the incentive-compatibility proof.

**The Shifting mechanism (outline)**:

*Divide the rectangles to $\log n$ classes according to their heights (e.g., all the rectangles up to height of 2, up to 4, 8 etc.). Run the following algorithm on each class, and output the solution for the class with the best solution:*

1. *Superimpose a set of parallel horizontal lines (distances between lines are class specific). Ignore all the rectangles that are intersecting these lines*
2. *For each slab (area between consecutive lines), project all the rectangles in it on the x-axis. Find the set of disjoint projections that maximize the sum of values (using dynamic programming). Sum up the results of all slabs.*

[4]The running time is polynomial in the size of the input representation, i.e., in $\sum_{i=1}^{n} n_i$.

3. *Shift all lines, and perform the above stages for every shift. Output the solution of the shift that obtained the highest result.*

We now illustrate how the algorithms that run on any two shifts (in stage 2) are monotonically crossing. We use the following trick. For every agent $i$ with OR-linear valuation function $v_i$, we create a set of dummy single-minded agents for each one of his desired bundles $S_j$ with a value of $\alpha_j \theta_i$. We then run the mechanism of (Babaioff & Blumrosen 2006) on the dummy agents, and allocate the winning rectangles to the original agents accordingly.

*Proof.* (Of Proposition 2 – sketch)

The atomic algorithms are those described at stage 2 of the above algorithm – that run for every particular shift of the horizontal lines and for each class of rectangles. The mechanism takes the maximum over such atomic algorithms. The main observation here is that fixing the types $\theta_{-i}$ of the other agents, the output (in terms of social welfare) of each of these atomic algorithms is a piecewise linear continuous function of $\theta_i$.[5] Once we show this, each pair of these algorithms will be monotonically crossing, since continuous piecewise-linear functions cross only when one curve has a higher slope than the other. A higher slope implies a "better" alternative for agent $i$ (according to the order $\succ_i$ implied by the single-crossing structure), and therefore this crossing must be monotone (the social-value curves have, in our case, the same slopes as the valuation functions). □

## Single-Parameter Combinatorial Auctions

We now show by example how the MAX-safe property easily derives the bitonicity property introduced in (Mua'lem & Nisan 2002).

Consider again the combinatorial-auction instance in Figure 1. In such combinatorial auctions, each agent has winning alternatives (here, with 45 degree slope) or losing alternatives (with a slope of zero). We saw earlier that if the algorithm shifts from choosing one winning alternative to a winning alternative with a lower social value at some particular profile of types, then this may cause a non-monotone potential crossing point (e.g., moving from $a_7$ to $a_5$ by Algorithm 1 in the point $\theta_i^{**}$). On the other hand, it is easy to see that moving upwards between winning alternatives (like Algorithm 2 moves from $a_6$ to $a_7$) cannot create non-monotone potential crossing points. Similarly, moving downwards between losing alternatives (like from $a_2$ to $a_1$) is safe, while moving upwards between losing alternatives is not. It follows that an algorithm that moves downwards (w.r.t. the social value) while the particular agent loses, and moves upwards when this agent wins, will be MAX-safe. This is exactly the definition of bitonicity.

Theorem 2 can similarly identify similar conditions for more complicated environments, e.g., for OR-linear valuations, and even for non welfare maximizing settings, like minimizing the makespan in job scheduling.

**Acknowledgments** I thank Shahar Dobzinski and Noam Nisan for helpful discussions.

[5] To see the intuition behind this argument, consider an agent that does not win a particular rectangle in a particular shift. Since the *optimal* allocation is calculated for the projections of the rectangles on the x-axis in each slab, for high enough $\theta_i$, the optimal result in this slab can be either achieved with this rectangle or with another allocation in which this rectangle still loses. For higher $\theta_i$'s, this rectangle will clearly win. Therefore, the transition from an outcome where this rectangle is losing to an outcome where it wins is continuous.

## References

Andelman, N.; Azar, Y.; and Sorani., M. 2005. Truthful approximation mechanisms for scheduling selfish related machines. In *STACS*, 69–82.

Babaioff, M., and Blumrosen, L. 2006. Computationally-feasible auctions for convex bundles. *Games and Economic Behavior, to appear*.

Babaioff, M.; Lavi, R.; and Pavlov, E. 2005. Mechanism design for single-value domains. In *20th National Conference on Artificial Intelligence (AAAI'05)*, 241–247.

Blumrosen, L., and Feldman, M. 2006. Implementation with a bounded action space. In *Proceedings of the 7th ACM conference on Electronic commerce*, 62–71.

Briest, P., and Krysta, P. 2006. Single-minded unlimited supply pricing on sparse instances. In *SODA*, 1093–1102.

Dobzinski, S.; Nisan, N.; and Schapira, M. 2005. Approximation algorithms for combinatorial auctions with complement-free bidders. In *The 37th ACM symposium on theory of computing*, 610–618.

Edlin, A. S., and Shannon, C. 1998. Strict single crossing and the strict Spence-Mirrlees condition: a comment on monotone comparative statics. *Econometrica* 68(6):1417–1425.

Goldberg, A. V.; Hartline, J. D.; Karlin, A. R.; Saks, M.; and Wright, A. 2006. Competitive auctions. *Games and Economic Behavior* 55(2):242 – 269.

Hermalin, B. E. 2005. Lecture notes in economics. University of California, Berkeley.

Kao, M.-Y.; Li, X.-Y.; and Wang, W. 2005. Towards truthful mechanisms for binary demand games: a general framework. In *Proceedings of the 6th ACM conference on Electronic commerce*, 213–222.

Khanna, S.; Muthukrishnan, S.; and Paterson., M. 1998. On approximating rectangle tiling and packing. In *the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 384–393.

Lehmann, D.; O'Callaghan, L. I.; and Shoham, Y. 2002. Truth revelation in approximately efficient combinatorial auctions. In *JACM 49(5)*, 577–602.

Mookherjee, D., and Reichelstein, S. 1992. Dominant strategy implementation of bayesian incentive compatible allocation rules. *Journal of Economic Theory* 56(2):378–399.

Motwani, R., and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press.

Mua'lem, A., and Nisan, N. 2002. Truthful approximation mechanisms for restricted combinatorial auctions. In *AAAI-02*.

Nisan, N., and Ronen, A. 2001. Algorithmic mechanism design. *Games and Economic Behaviour* 35:166 – 196. A preliminary version appeared in STOC 1999.

Segal, I. 2003. Optimal pricing mechanisms with unknown demand. *American Economic Review* 93(3):509–529.