An Ironing-Based Approach to Adaptive Online Mechanism Design in Single-Valued Domains

David C. Parkes and **Quang Duong**
School of Engineering and Applied Sciences, Harvard University
Cambridge, MA USA {parkes,qduong}@eecs.harvard.edu

Abstract

Online mechanism design considers the problem of sequential decision making in a multi-agent system with self-interested agents. The agent population is dynamic and each agent has private information about its value for a sequence of decisions. We introduce a method ("ironing") to transform an algorithm for online stochastic optimization into one that is incentive-compatible. Ironing achieves this by canceling decisions that violate a form of monotonicity. The approach is applied to the CONSENSUS algorithm and experimental results in a resource allocation domain show that not many decisions need to be canceled and that the overhead of ironing is manageable.

Introduction

Mechanism design (MD) studies the problem of decision making in multi-agent systems with rational, self-interested agents each with private information about their valuations for different decisions. The central problem in MD is to provide *incentive-compatibility* (IC). A mechanism is incentive compatible if the optimal strategy for an agent is to report its private information truthfully.

Online MD extends the methods of MD to dynamic environments, in which the agent population is dynamically changing, decisions must be made across time, and there can be uncertainty about the set of feasible decisions in future periods. Example applications include:

- Selling last-minute theater tickets. Customers arrive to a ticket booth with a value for some number of tickets and need a decision before some deadline.
- Selling network access. Customers arrive into a coffee house with a value for access to a shared WiFi base station for some contiguous period of time and before some deadline.

This paper addresses a fundamental challenge in online MD: *how can one use general-purpose algorithms for online stochastic optimization to make decisions while also providing incentive-compatibility*? In particular, we want to be able to achieve IC without requiring either that the algorithm computes the optimal policy or that the algorithm has access to a correct probabilistic model of the environment.

To be concrete, consider the WiFi problem. As customers arrive and depart they can choose to bid for WiFi access. The current bids along with any current allocations form the state of the mechanism at any point in time. Suppose an online algorithm is used to propose allocation decisions that maximize expected social welfare given a probabilistic model of the bid arrival process. We seek *dominant-strategy* IC. This means that it is optimal for a customer to report her true valuation for WiFi access and immediately upon arrival, whatever the bids of other customers and for all possible futures. This simplifies the bidding problem by removing the need for wasteful counter-speculation.

The basic idea is very simple. Whereas the online algorithm may not have the required properties to allow for IC, we introduce a method ("ironing") to transform the algorithm into one with these properties. Ironing[1] achieves this by *canceling* any allocation decision that violates a form of monotonicity. We will see that the *quid pro quo* for completely removing the strategic problem facing customers is that we must cancel only a small number of decisions (typically $< 1\%$).

To check for a violation of monotonicity, the ironing procedure needs to perform sensitivity analysis on the decisions of the online algorithm. Specifically: given a proposed allocation it must be possible to construct the sequence of decisions that would have been made for an alternate bid that a provisionally allocated customer could have made. For this to be possible we need the property of *uncertainty independence* (UI). UI requires that the realization of uncertain events, such as bids and the supply of resources, is independent of decisions. Without UI, the mechanism could have no knowledge of the events of this kind that would have occurred subsequent to the first period in which a different decision would have been made.

[1]The term "ironing" is inspired by the ironing procedure adopted by Myerson (1981) to make the allocation rule of a revenue-maximizing (offline) auction monotone and thus truthful. Ironing is an analytic technique in that paper. Here it is performed online and made computational.

Here we have good news. First, UI appears to be well motivated in many resource allocation domains. Second, UI facilitates scalable computational methods for online stochastic combinatorial optimization (OSCO) (Van Hentenryck & Bent 2006). UI allows one to combine sample approximation methods with systematic search algorithms when evaluating which decision to make in some time period.

We illustrate the use of ironing by application to the CONSENSUS (**C**) algorithm for OSCO (Bent & Van Hentenryck 2004). **C** is well-suited to ironing because its use of combinatorial optimization allows for tractable sensitivity analysis. When checking for violation of monotonicity it is possible to compute a small number of candidate bid values at which decisions could have changed. Experimental results in a simulated WiFi domain show that not many decisions need to be canceled and that the overhead of ironing is manageable.

Related Work

Current results in online MD fall into two broad categories (see Parkes (2007) for a survey).

First, special-purpose online algorithms have been developed that provide dominant-strategy IC on specific problems (Lavi & Nisan 2000; Hajiaghayi, Kleinberg, & Parkes 2004). These algorithms are designed for *worst-case* performance and assume an adversary whose objective is to minimize performance.

Second, a dynamic generalization of the Vickrey-Clarke-Groves (VCG) mechanism can be used to achieve IC when the goal is to maximize *expected* social welfare (Parkes & Singh 2003). This VCG-based approach has two limitations: (a) it requires a correct probabilistic model of the multi-agent environment and an optimal (or ϵ-optimal) algorithm for online stochastic optimization; (b) it achieves Bayes-Nash IC but not dominant-strategy IC, so that truthful bidding is only optimal when every agent plays this equilibrium and given common priors.

We bridge these two categories by achieving dominant-strategy IC while still adopting a probabilistic model to inform the decision making of the mechanism. We require neither a *correct* probabilistic model, nor that the mechanism's decision policy is optimal.

Awerbuch et al. (2003) also modify online algorithms to make them monotonic, although in their work this is achieved by modifying the *input* to the algorithm rather than the *output*. Unlike our work, they adopt an adversarial rather than probabilistic framework and critically rely on finite-horizon problems where the decision problem gets more and more constrained over time. This ensures that (competitive) online algorithms are "almost" truthful. A recent paper on "incremental mechanism design" (Conitzer & Sandholm 2007) is in the same spirit of achieving truthfulness through modifying an algorithm, albeit for offline mechanisms. Pai and Vohra (2006) seek revenue-optimality in dynamic auctions but are unable to assure monotonicity through dynamic programming.

The Model

Consider discrete time periods $T = \{1, 2, \dots, \}$, indexed by t and possibly infinite. Notation $[t_1, t_2]$ denotes $\{t_1, \dots, t_2\}$. A mechanism makes a sequence of decisions $k = (k^1, k^2, \dots,)$, with decision $k^t \in K^t$ made in period t, where K^t denotes the set of feasible (single-period) decisions. We use K to denote the domain of all possible single-period decisions.

Example 1. *One can model the WiFi problem with a decision k^t in period t that defines an allocation of resources for some period of time to a subset of unallocated customers. The feasible decisions in period t depend on resources that are already committed.*

Each agent i has a private *type* $\theta_i \in \Theta_i$. The type of an agent defines its value for sequences of decisions and will be associated with both an "arrival" and a "departure" period. We consider *direct-revelation* mechanisms in which each agent is restricted to make a single claim about its type. Let θ^t denote the set of agent types reported in period t. The *state*, $h^t = (\theta^{1..t}, k^{1..t-1})$, of a mechanism in period t captures all information relevant to its decision, i.e. the history of all reported types and decisions made up until the current period.

An **online mechanism** $M = (\pi, x)$ defines *decision policy* $\pi = \{\pi^t\}^{t \in T}$ and *payment policy*, $x = \{x^t\}^{t \in T}$, with decision $k^t = \pi^t(h^t, \omega^{1..t})$ and payment $x_i^t(h^t, \omega^{1..t}) \in \mathbb{R}$ collected from each agent $i \in n^t(\theta^{1..t})$, in period t. Notation $n^t(\theta^{1..t})$ is used to denote the set of agents that are present in period t. The decision policy may itself be stochastic, modeled here by adopting $\omega = (\omega^1, \omega^2, \dots)$ to denote the stochastic events *internal* to decision policy π.

Single-Valued Preferences

Agents have quasi-linear utility functions, so that each agent seeks to maximize the expected difference between its value for decisions and its payment. We study domains in which agents have *single-valued* (SV) preferences (Babaioff, Lavi, & Pavlov 2005). A SV preference domain is defined by $\mathcal{L}_i$, the domain of *interesting sets* (of decisions), and a partial order $\succeq_L$ on interesting sets $L' \in \mathcal{L}_i$, where $L' \subseteq K$. An agent has the same value for any decision in some interesting set.

Example 2. *In the theater tickets problem, SV preferences allow for a customer that has the same value for a decision in an interesting set that defines an allocation of two tickets to any of three shows. We cannot have a customer with different values for different shows, or with a different value for different numbers of tickets to some show. In the WiFi problem, this allows for a customer that has the same value for getting online for any contiguous period of 20 mins (these allocations are "interesting") while she is present in the coffee house but does not allow for a customer with some value for being online for 5 mins and a different value for 20 mins.*

Formally, an agent's type, $\theta_i = (a_i, d_i, (r_i, L_i)) \in \Theta_i$, where $\Theta_i \subseteq T \times T \times \mathbb{R} \times \mathcal{L}_i$, defines value r_i for a

suitably timed decision in interesting set L_i. Periods a_i and d_i define the agent's *arrival* and *departure* and delineate the interval of time in which the agent values a decision. Agent i has valuation,

$$v_i(\theta_i, k) = \begin{cases} r_i, \text{ if } k^t \in <L_i, \succeq_L> \text{ for some } t \in [a_i, d_i] \\ 0 \text{ , otherwise,} \end{cases} \quad (1)$$

on a sequence of decisions k. Here, we adopt

$$<L_i, \succeq_L> = \bigcup_{L' \succeq_L L_i, L' \in \mathcal{L}_i} L', \quad (2)$$

to denote the set of decisions that are either in L_i or in some interesting set that is "better" (with respect to $\succeq_L$) than L_i. A decision that meets the criteria of Eq. (1) is said to *satisfy* the agent.

We require that interesting sets are *pairwise disjoint*, with $L'_i \cap L''_i = \emptyset$ for all $L'_i, L''_i \in \mathcal{L}_i$. This ensures well-defined (single-valued) value semantics, in that there always exists a single type θ_i that defines an agent's valuation via Eq. (1). Without this, a decision for which agent i has value could be included in two sets, L'_i and L''_i, for which neither $L'_i \succeq_L L''_i$ or $L''_i \succeq_L L'_i$.

Example 3. *In the theater ticket problem, period a_i models the earliest period of time at which the customer can arrive at the ticket booth and period d_i models the latest period of time at which the customer can check back to pick up tickets. A partial order can be defined to satisfy free-disposal so that a customer that demands 2 tickets for one of two shows is also happy with more than 2 tickets. For disjoint interesting sets, we need, for example that each customer is either indifferent between shows A and B* or *indifferent between shows C and D. We cannot model a domain in which some customers are indifferent between shows A and B and some are indifferent between shows B and C. The disjoint interesting sets property is trivially satisfied in the standard "single-minded" model in which each agent wants exactly one bundle of goods, for example if every customer is interested in exactly one show.*

For a SV domain to be **well-defined** (and suitable for ironing) we require it to satisfy the following three properties:

Property A1: Lower-availability. Suppose decision k^t is feasible in period t and satisfies agents $W \subseteq n^t(\theta^{1..t})$, perhaps with $k^t \in < L_i, \succeq_L > \backslash L_i$ for some $i \in W$. Then, there exists a feasible decision k' in the current period such that: (a) $k' \in L_i$ for all $i \in W$, and (b) k' does not prevent any future decisions that would be possible following decision k^t.

Property A2: Cancelability. Suppose decision k^t is feasible in period t and satisfies agents $W \subseteq n^t(\theta^{1..t})$. Then, for any $W' \subset W$ there exists a feasible decision k' in the current period such that: (a) k' satisfies W' but no agent in W but not W', and (b) k' does not prevent any future decisions that would be possible following decision k^t.

Property A3: Uncertainty Independence. The realization of (true) agent types in future periods does not depend on current or previous decisions.

Property A1 is useful because it allows one to focus WLOG on *exact* policies, in which any decision k^t made in period t that satisfies agent i is in interesting set L_i and not in some $L' \succ_L L_i$. Property A2 is important because ironing needs to be able to modify a proposed decision to cancel the positive impact on any subset of provisionally satisfied agents.

Example 4. *Properties A1 and A2 are satisfied in the theater ticket problem when the partial order is defined to respect free disposal. For example, suppose that L_i defines all allocations that allocate to the customer 2 tickets to show A or show B and an interesting set $L' \succ_L L_i$ defines allocations that allocate to the customer 2 tickets to* one of *show A or show B and 1 ticket to the other show. For Property A1, note that if an allocation is made in L' it is also feasible to just allocate the customer 2 tickets in one of the shows and discard the ticket to the other show. For Property A2, note that it is always possible to simply decide not to allocate tickets to a customer and cancel a decision.*

As discussed earlier, Property A3 (UI) is important both for ironing to be able to check for violations of monotonicity and also for computational tractability in interesting domains.

Example 5. *UI is satisfied in the theater ticket problem when the arrival of customers is independent of whether or not there remain tickets to shows. When selling WiFi access, UI requires that the arrival of customers is independent of particular decisions made about how to allocate WiFi.*

Truthfulness and Monotonicity

We now provide a characterization of decision policies that can be truthfully implemented in well-defined SV domains. To keep the presentation simple[2] we will assume restricted misreports, specifically *no early-arrival and no late-departure misreports* so that misreport $\hat{\theta}_i = (\hat{a}_i, \hat{d}_i, (\hat{r}_i, \hat{L}_i))$ must satisfy $a_i \leq \hat{a}_i \leq \hat{d}_i \leq d_i$.

Example 6. *To motivate restricted misreports in the theater tickets problem, suppose that it is too costly for the customer to arrive at the ticket booth before a_i and too costly to return later than d_i and suppose the mechanism will not release the tickets until an agent's reported departure. In the WiFi access problem, suppose that a_i is the first period at which the customer realizes her need for Internet access, and that when bidding the customer is required to make a deposit that is only returned if she is present upon her reported departure.*

[2] A more complicated monotonicity-based characterization is available for dominant-strategy IC in which only one of no early-arrival or no early-departure can be assumed; see Hajiaghayi *et al.* (2005).

For dominant-strategy IC, we require that truthful reporting is the optimal strategy for an agent whatever the reports of other agents and whatever the realization of stochastic events ω. (The OSCO algorithms will use random sampling to predict possible future arrivals of bids.) We call this property of a mechanism *strong truthfulness*.

Adopt shorthand $\pi_i(\theta_i, \theta_{-i}, \omega) \in \{0, 1\}$ to indicate whether policy π makes an interesting decision for agent i in some period $t \in [a_i, d_i]$, given reports $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots)$ by other agents and stochastic events ω. Define the *critical-value* for agent i with type $\theta_i = (a_i, d_i, (r_i, L_i))$ as $v^c_{(a_i,d_i,L_i)}(\theta_{-i}, \omega) = \min r'_i$ s.t. $\pi_i((a_i, d_i, (r'_i, L_i)), \theta_{-i}, \omega) = 1$, or ∞ if no such r'_i exists. This is the smallest value that agent i could report and still receive an interesting decision, all else unchanged.

Strong monotonicity is defined with respect to a partial order on agent types,

$$\theta_1 \preceq_\theta \theta_2 \equiv (a_1 \geq a_2) \wedge (d_1 \leq d_2) \wedge (r_1 \leq r_2) \wedge (L_1 \succeq_L L_2)$$

In words, $\theta_1 \preceq_\theta \theta_2$ if θ_1 has a tighter arrival-departure interval, a lower value and a higher interesting set with respect to partial order $\succeq_L$ than θ_2.

Definition 1 (strongly monotonic)**.** *Stochastic policy π is strongly monotonic if $(\pi_i(\theta_i, \theta_{-i}, \omega) = 1) \wedge (r_i > v^c_{(a_i,d_i,L_i)}(\theta_{-i}, \omega)) \Rightarrow \pi_i(\theta'_i, \theta_{-i}, \omega) = 1$ for all $\theta'_i \succ_\theta \theta_i$, for all θ_{-i}, all $\omega \in \Omega$, where $\theta_i = (a_i, d_i, (r_i, L_i))$.*

For example, in the WiFi access problem, strong-monotonicity requires that when a decision policy allocates to a customer in some period, it would also have allocated to the customer (in some period) if the customer had an earlier arrival, a later departure, a higher value, or when asking for less resources.

Lemma 1. *Agent i receives no value from the decisions of an exact policy unless it reports some set $L'_i \succeq_L L_i$.*

Proof. The policy is exact, and thus will only make a decision in L'_i. The result then follows from the definition of value in a SV domain since agent i will have no value for a decision unless $k^t \in L'_i \succeq_L L_i$. □

Given this lemma the following theorem follows as a simple extension to results in Hajiaghayi et al. (2005).[3] A mechanism is individual-rational (IR) if no agent achieves negative utility in equilibrium.

Theorem 1. *An exact policy π can be strongly truthfully implemented in an IR mechanism that does not pay unallocated agents, and in a SV domain with no-early arrival and no-late departure misreports, if and only if the policy is strongly monotonic.*

[3]Babaioff et al. (2005) deal with SV domains in a static environment and require the existence of "distinguishing minimal elements" for each interesting set. Roughly speaking, these are decisions that are only of value to an agent with that particular interesting set of decisions. In our set up, disjointness of interesting sets means that *every* decision in set L_i is a distinguishing minimal element in this sense.

The payment policy that, when coupled with a strongly monotonic policy provides truthfulness, is defined to *collect the critical-value from an agent upon departure* when the agent is allocated in some period, with zero payment for an unallocated agent.

Our ironing method will enforce a stronger variant on monotonicity, that we refer to as *anytime monotonicity*. Let $\pi_i^{\leq t}(\theta_i, \theta_{-i}, \omega) \in \{0, 1\}$ denote whether or not an interesting decision is made for agent i *by period* t. Given this, anytime monotonicity is defined as:

Definition 2 (anytime monotonicity)**.** *Stochastic policy π is anytime monotonic if $(\pi_i^{\leq t}(\theta_i, \theta_{-i}, \omega) = 1) \wedge (r_i > v^c_{(a_i,d_i,L_i)}(\theta_{-i}, \omega)) \Rightarrow \pi_i^{\leq t}(\theta'_i, \theta_{-i}, \omega) = 1$ for all $\theta'_i \succ_\theta \theta_i$, for all θ_{-i}, all $\omega \in \Omega$, and all $t \in [a_i, d_i]$.*

This strengthens monotonicity to require that it holds in any period and not just upon departure. This is important to facilitate ironing, which must be based on the information available when a decision is about to be made for an agent.

Example: Failure of Monotonicity

But, will (approximately) optimal policies be automatically monotonic? Consider a domain with 3 non-expiring units of an identical good to allocate in $T = 2$ periods. Each agent i demands some quantity q_i of units. We present an example to show failure of monotonicity with respect to value.[4] Denote an agent's type $(a_i, d_i, (r_i, q_i))$ and consider the following types for agents 1 and 2 in period 1 and probabilistic information about the type of an agent that will arrive in period 2:

period 1	$A_1 : (1, 1, (5, 1))$, $A_2 : (1, 2, (500, 2))$
period 2	$A_3 \sim (2, 2, (1000, 3))$ with high prob
	$(2, 2, (5000, 1))$ with low prob

The optimal policy will not make a decision about A_2 until period 2. Furthermore, it is not hard to see that it will also choose not to allocate to A_1. The marginal value of saving the third unit is high, since it is likely that type $(2, 2, (1000, 3))$ will arrive in period 2. Now, suppose that agent A_3 with type $(2, 2, (5000, 1))$ arrives in period 2. The policy will allocate A_2 and A_3 in period 2. Consider now that agent 2's type is $(1, 2, (\mathbf{1000}, 2))$. The optimal policy will now allocate to A_1 in period 1 because the marginal value of saving the third unit is low, since it cannot allocate to both A_2 and another type $(2, 2, (1000, 3))$ anyway and it is unlikely that type $(2, 2, (5000, 1))$ will arrive. Then the unlikely event occurs, A_3 arrives and the policy allocates to A_3 but not A_2 in period 2. We see a failure of monotonicity.

The Ironing Procedure

An *ironing procedure* is a "wrapper" around a decision policy π that is able to modify the decision k^t proposed

[4]We also have counterexamples for monotonicity with respect to arrival, quantity, and departure, and for the expiring goods domain considered in the experimental section.

by π in any period by canceling the effect of the decision to some number of agents.

Given policy π, we say that we *iron* the policy to construct ironed policy $\breve{\pi}$. Let $t_i^\pi(\theta_i, \theta_{-i}, \omega) \in T \cup \{\infty\}$ denote the period (∞ if none exists) in which an interesting decision is made for agent i given policy π. Sometimes this is abbreviated to $t_i^\pi(\theta_i)$.

Definition 3 (ironing procedure)**.** *Consider period t and exact policy π. Initialize ironed decision, $\breve{k}^t := k^t = \pi^t(h^t, \omega^{1..t})$. For each $i \in n^t(\theta^{1..t})$ for which $\breve{k}^t \in L_i$, and given ω, check*

$$t_i^\pi(\theta_i'', \theta_{-i}, \omega) \leq t_i^\pi(\theta_i', \theta_{-i}, \omega), \tag{3}$$

for all $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$. If this fails, then modify $\breve{k}^t$ to cancel the effect of the decision to agent i.

Given a proposed decision k^t by policy π in some period, the ironing method checks for anytime monotonicity for all agents that are satisfied by decision k^t. Eq. (3) requires that the policy would have satisfied the agent for all higher types $\theta_i'' \succeq_\theta \theta_i$, and in particular with the period in which this occurs getting monotonically-earlier for all $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$. If this is not verified then the positive effect of the current decision to agent i is canceled. The result is modified decision $\breve{k}^t$.

Theorem 2. *Ironed policy $\breve{\pi}$ satisfies anytime monotonicity, and thus is strong monotonic and is feasible in a SV domain that satisfies Properties A2 and A3.*

Proof. If $\breve{\pi}_i^{\leq t}(\theta, \omega) = 1$ then $t_i^\pi(\theta_i'') \leq t_i^\pi(\theta_i') \leq t$ for all $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$, and thus $\breve{\pi}_i^{\leq t}(\hat{\theta}_i, \theta_{-i}, \omega) = 1$ for all $\hat{\theta}_i \succeq_\theta \theta_i$ since $t_i^\pi(\hat{\theta}_i) \leq t$ and Eq. (3) holds for all $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$ and thus for all $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \hat{\theta}_i \succeq_\theta \theta_i$. Ironing is feasible because Property A2 allows decisions to be canceled and Property A3 makes it possible to know what would have happened for alternate types. □

Example 7. *Here is a simple example to help to understand why we must verify **monotonically**-earlier allocations for $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$. Suppose we are in period 3 and policy π proposes to allocate to agent i. Upon checking sensitivity information, we find that agent i would be allocated in period 2 and then period 3 as we trace higher types $\theta_i \prec_\theta \theta_i' \prec_\theta \theta_i''$ (and thus monotonicity.) Yet, if we allowed this decision in period 3 then $\breve{\pi}$ would not be anytime monotonic because agent i reporting type θ_i' would not survive ironing in period 2 because in that period there would be no evidence available that type $\theta_i'' \succ_\theta \theta_i'$ will be allocated.*

It is important to realize that the algorithm ($\mathcal{A}$) used to define the (un-ironed) decision policy π is ignorant of the existence of the ironing procedure. The algorithm $\mathcal{A}$ has no information about whether or not a decision is canceled, and updates its local state as if the decision proceeded as planned. An agent that is provisionally satisfied in period t by $\mathcal{A}$, but for which the effect of the decision is canceled, ultimately goes unsatisfied. Moreover, in a resource allocation domain the resources (e.g. tickets, WiFi access) go wasted.

We also have a counterpart to Theorem 2 that shows that these decisions *must* be canceled:

Theorem 3. *Given policy π with arbitrary future behavior, an ironing procedure can only ensure that ironed policy $\breve{\pi}$ is monotonic by canceling all decisions for which Eq. (3) is not satisfied.*

Proof. Suppose an interesting decision to agent i is allowed in period t despite some $\theta_i' \succeq_\theta \theta_i$ for which $t_i^\pi(\theta_i') > t$. But now there is no proof of monotonicity, i.e. the policy need not allocate an agent with type θ_i' in some future period. Also, suppose $\pi_i^{\leq t}(\theta_i', \theta_{-i}, \omega) = 1$ for all $\theta_i' \succeq_\theta \theta_i$ but $t_i^\pi(\theta_i'') > t_i^\pi(\theta_i')$ for some $\theta_i'' \succeq_\theta \theta_i' \succeq_\theta \theta_i$. Consider an agent with report θ_i' and period $t' = t_i^\pi(\theta_i')$. Here, $\pi_i^{\leq t'}(\theta_i'', \theta_{-i}, \omega) = 0$ for some $\theta_i'' \succeq_\theta \theta_i'$ and the ironing procedure would need to cancel the decision to ensure monotonicity, as above. □

We now give our main theorem. Say that an adaptive, online algorithm (an example is provided in the next section) satisfies *delayed learning* if the information about an agent's reported type is not used for learning until the agent is no longer active.

Theorem 4. *An adaptive, online algorithm that computes an exact policy, coupled with ironing and delayed learning can be truthfully-implemented in well-defined SV domains and with no early-arrival and no late-departure misreports.*

Proof. (sketch) Delayed learning ensure that an agent's report cannot affect the model and thus have a new way to influence the decisions made by the algorithm. Coupled with ironing, we have strong monotonicity by Theorem 2 (and see that ironing is feasible since the domain is well-defined) and strong-truthfulness by Theorem 1. □

Note that critical-value payments are computed with respect to the ironed policy. Determining this requires reasoning about the ironing procedure. We make this explicit in the next section, in reference to the CONSENSUS algorithm.

Discussion

Ironing is performed in a way that is transparent to the algorithm $\mathcal{A}$ that is used to determine policy π. This avoids a "ripple effect," with cascading re-optimization when ironing.

Suppose this was not the case and that the state of the algorithm is updated to reflect whether or not a proposed decision is canceled. Consider now that we are checking for a violation of monotonicity given a proposed decision to agent i in some period $t = a_i + 1$. When checking the effect of types $\theta_i' \succeq_\theta \theta_i$ on the decision made in period a_i we would need to consider not only the effect of an alternate type θ_i' on the decision itself, but also the effect on whether or not the proposed decision is canceled. To avoid this, we run algorithm $\mathcal{A}$ in "open loop" and without feedback about ironing, so that its future decisions are unaffected by whether or not its current decision is canceled.

Ironing applied to CONSENSUS

To illustrate the framework we apply ironing to the CONSENSUS (**C**) algorithm (Bent & Van Hentenryck 2004). **C** is a general-purpose, online algorithm for stochastic optimization in environments that satisfy UI. **C** is run in each period t and makes a decision just for the current period. We often refer to a satisfying decision as an "allocation" because we apply this method to a resource allocation domain.

function CONSENSUS(h^t, T_l, J)
  votes(k):=0 for each $k \in K$
  $s(j)$:=GetSample(t, T_l) for each $j \in \{1, \ldots, J\}$
  for $j \in \{1, \ldots, J\}$ **do**
    o:= Opt(h^t,$s(j)$)
    votes(o^t):=votes(o^t)+1
  end for
  return $\arg\max_k$ votes(k)
end function

Figure 1: The CONSENSUS algorithm defined in state h^t, with look-ahead horizon T_l and J scenarios.

C (Figure 1) generates J scenarios for the future arrival of agent types from the current period t until period $t + T_l$, where T_l is the *look-ahead horizon*. Let $j \in \{1, \ldots, J\}$ index a scenario and denote an individual scenario $s(j)$. A scenario is a set of agent types with arrival times in interval $[t, t+T_l]$. Given these scenarios, **C** solves an optimization problem for each j, to determine what would be the optimal sequence of decisions if the future looked exactly as predicted in $s(j)$.

The solution to each one of these optimization problems generates a *vote* for *the decision that would be optimal in the current period if the future did indeed look like* $s(j)$. **C** ultimately selects the decision in the current period that receives the most votes. As such, **C** is elitist.[5] Note that **C** exploits UI by sampling trajectories before making decisions.

As building blocks the **C** algorithm requires: (a) a method GetSample(t, T_l) to generate a scenario, which is a sample of a possible realization of types θ in periods $[t+1, t+T_l]$; (b) a method Opt$(h^t, s(j))$ to solve an offline, complete-information optimization problem to determine an optimal sequence of decisions given a scenario. **C** can be made adaptive by adopting *historical sampling*, with a scenario generated by sampling from the observed history (Van Hentenryck & Bent 2006). We adopt this approach in our work.

Value Ironing

We first explain ironing with respect to value. The ironing procedure needs to trace how the decisions made by **C** would change in response to the report of a higher value by a provisionally allocated agent. If the allocation occurs in period $t*$, then a report of a higher value could have a potential impact on the decision made by

[5]Other "anticipatory algorithms" have also been defined (Van Hentenryck & Bent 2006) but we choose to study **C** because of its simplicity.

function V-IRONING(h^{t*}, k^{t*}, i)
  $\check{r}_i$:=r_i, $\check{t}_i$:=$t*$, $\check{h}^t := h^t \ \forall t \in [a_i, t*]$, FAIL:=false
  Construct DM$_i^t(\check{h}^t)$ for each $t \in [a_i, t*]$
  while ($\min_{t\in[a_i,\check{t}_i]} r_i^{c+}(\check{h}^t, \check{r}_i) < \infty$) & (¬FAIL) **do**
    $r'_i := \min_{t\in[a_i,\check{t}_i]} r_i^{c+}(\check{h}^t, \check{r}_i)$, set t'_i to corresponding period (break ties earlier)
    update DM$_i^t(\check{h}^t)$ and $\check{h}^t$ for $t \in [t'_i + 1, \check{t}_i]$ to reflect value r' and new decision in t'_i
    if $t_i^\pi((a_i, d_i, (r'_i, L_i)), \theta_{-i}, \omega) > \check{t}_i$ **then**
      CANCEL decision for agent i, FAIL:=true
    else
      $\check{t}_i := t_i^\pi((a_i, d_i, (r'_i, L_i)), \theta_{-i}, \omega)$, $\check{r}_i := r'_i$
    end if
  end while
end function

Figure 2: The V-IRONING procedure for a policy computed with the CONSENSUS algorithm.

C in each period $t \in [a_i, t*]$. In particular, we must check that $t_i^\pi(r''_i) \le t_i^\pi(r'_i) \le t*$ for all $r''_i \ge r'_i \ge r_i$ (supressing θ_{-i}, ω and the irrelevant aspects of agent i's type from our notation.)

Suppose that Opt$(h^t, s(j))$ is defined to maximize the total value of the decisions in periods $[t, t + T_l]$. In determining the sensitivity of **C** to different values we must find the values for agent i for which the votes in this period would have changed.

In particular, consider scenario j and let $r_i^c(h^t, j)$ denote the *minimal* value for which i is allocated in the solution to Opt$(h^t, s(j))$, and ∞ if there is no such value. Note that this is the value at which the solution to this particular *offline* problem would change. Sensitivity information for these offline problems leads to sensitivity information for the online problem. Let $o^t(r'_i, h^t, s(j))$ denote the decision voted by scenario $s(j)$ in period t given value r'_i.

Lemma 2. *The decision $o^t(r'_i, h^t, s(j))$ voted by scenario $s(j)$ in state h^t given value r'_i to agent i is constant for all $r'_i < r_i^c(h^t, j)$, and constant for all $r'_i \ge r_i^c(h^t, j)$ (breaking ties in the same way).*

Proof. Let $V(X)$ denote the total value of the optimal solution X to Opt$(h^t, s(j))$, where the solution is restricted to those that *do not* allocate to agent i. Let $V_{-i}(Y)$ denote the total value to agents $\neq i$ of the optimal solution Y to Opt$(h^t, s(j))$, where the solution is restricted to those that *do* allocate to agent i (and $V_{-i}(Y) = -\infty$ if no such solution exists.) For $v'_i < r_i^c(h^t, j)$, the solution to Opt$(h^t, s(j))$ is X and independent of v'_i. At $v'_i = r_i^c(h^t, j)$, we have solution Y and $V_{-i}(Y) + v'_i = V(X)$. The total value of solution Y (including the value to agent i) increases linearly with v'_i for values $v'_i \ge r_i^c(h^t, j)$, and by at least as much as the total value of any other solution. Therefore Y is the optimal solution for all $v'_i \ge r_i^c(h^t, j)$. □

Because of this we can compute break-point $r_i^c(h^t, j)$ by solving two optimization problems, with i forced in and out of the solution to Opt$(h^t, s(j))$. The union of the breakpoints over all scenarios defines possible values

at which the decision made by **C** (i.e., that with the most number of votes) in state h^t can change.

Figure 2 defines the value-ironing procedure. It tracks the new state $\breve{h}^t$ in periods $t \in [a_i, t*]$ as agent i's value is increased. An important data structure is the "decision map" for each period $t \in [a_i, t*]$, $\text{DM}_i^t(\breve{h}^t) = \{(r^{(m)}, k^{(m)})\}^{m \in \{0,\dots M\}}$, which defines the break-points in value to agent i for which $\pi^t(\breve{h}^t)$ changes; where $r^{(0)} := r_i$ and $(r^{(M)}, k^{(M)}) = (\infty, \_)$, and decision $k^{(m)}$ is made for all $r_i' \in [r^{(m)}, r^{(m+1)})$, with $M \leq J+1$ by Lemma 2. DM's are maintained for each period $t \in [a_i, t*]$ and used to track changes in the decisions of **C** as r_i increases. In the algorithm, $r_i^{c+}(\breve{h}^t, \breve{r}_i)$ denotes the smallest value $r_i' > \breve{r}_i$ at which the decision in period t will change.

Theorem 5. *Algorithm* V-IRONING *is a correct value-ironing procedure for the* CONSENSUS *algorithm.*

Correctness follows from Lemma 2, with the proof that the correct state of **C** is traced as r_i increases established by induction on current value $\breve{r}_i$.

A completely analogous algorithm can be used to compute the critical-value payment. In the departure period d_i for an allocated agent we initialize DM's for periods $t \in [t*, d_i]$ and values smaller than r_i. We find the smallest value for which: (a) agent i continues to be allocated, (b) the allocation period is monotonically-later as the agent's value increases. If some value break-point $\breve{r}_i$ is found for which the agent would still be allocated, but in some *earlier* period $\breve{t}_i$ than $r_i' = \breve{r}_i + \epsilon$, then an agent with type $\breve{r}_i$ would not be allocated by the ironed policy because the value-ironing check would fail when performed online in period $\breve{t}_i$.

Value and Departure Ironing

We briefly describe ironing with respect to both value and departure. For this, we require a *maximal patience*, $\Delta \in \{0, 1, \dots\}$, so that $d_i \leq a_i + \Delta$ for all types $\theta_i \in \Theta_i$. First we use the V-IRONING procedure with agent i's departure d_i substituted with every possible later departure $d_i' \in [d_i, a_i + \Delta]$ and establish (**B1**), $t_i^\pi(d_i', r_i'') \leq t_i^\pi(d_i', r_i')$ for all $r_i'' \geq r_i' \geq r_i$. If FAIL for any d_i' then CANCEL the decision. Else, (as a side effect of the procedure) construct *timing map*, $TM_i(d_i') := \{(r^{(m)}, t^{(m)})\}^{m \in \{0,\dots,M\}}$, with $r^{(0)} := r_i$ and $(r^{(M)}, t^{(M)}) = (\infty, a_i)$. This indicates the value break-points for which $t_i^\pi(\theta_i)$ changes given departure d_i': agent i would be allocated in period $t^{(m)}$ for value in interval $[r^{(m)}, r^{(m+1)})$. Second, we check monotonicity with respect to misreports of both value and departure. For this, we establish (**B2**), that $t_i^\pi(d_i'+1, r_i') \leq t_i^\pi(d_i', r_i')$ for all $r_i' \geq r_i$ and all $d_i' \in [d_i, a_i + \Delta - 1]$. The break-points in the TM's provide enough information for this check. (**B1**) and (**B2**) imply (anytime) monotonicity with respect to value and departure.

Experimental Results: Expiring Goods

We model the WiFi allocation problem. There are multiple, expiring units of an identical good available in each period. Each unit represents a single channel. Each bidder demands some number of channels q_i for some contiguous period of time l_i and have value r_i. We experiment with value and departure ironing and leave full ironing, that is including also quantity and arrival ironing for future work.

We consider $T = 100$ periods, $S = 5$ channels, number of arrivals in each period uniform on $\{1, \dots, n_{\max}\}$ for $n_{\max}$ varying between 2 and 10, r_i uniform on $(0, 7]$, q_i uniform on $\{1, 2, 3\}$, duration δ_i (i.e. departure-arrival+1) uniform on $\{1, \dots, 7\}$, and length of request l_i uniform on $\{1, \dots, \delta_i\}$.[6]

We compare the **C** algorithm, with look-ahead horizon $T_l = 10$ and number of scenarios $J = 50$ with a **tree-sampling** algorithm for planning in MDPs (Kearns, Mansour, & Ng 1999), defined with look-ahead horizon 4 and sample-width 6.[7] Both algorithms use historical sampling and are adaptive. CPLEX[8] is used to solve *Opt* in the **C** algorithm, with tie-breaking adopted to prefer allocations of items about to expire. All results are averaged over 10 trials.

Figure 3 (a) illustrates the value of **C**, **C**+V-Ironing and **C**+V-D-Ironing (value and departure ironing), with total value normalized with respect to the value from an optimal offline solution computed with perfect hindsight. We also plot the (normalized) performance of **C** on the first 20 time periods. The poor performance, relative to that over the entire 100 periods, shows that learning is effective.

The value of the ironed policies is within 0.1% of **C**. Most significantly, we see that while ironing was required, the ironing frequency is very low, with less than 0.5% (0.8%) of allocations canceled with V-Ironing (V-D-Ironing), and the maximal ironing occurring at $n_{\max} = 9$ ($n_{\max} = 6$). The performance of tree-sampling is much worse than that of **C** because of its limited look ahead and sample width. Revenue from **C**+V-D-Ironing (not plotted) varied from a normalized value of around 0.1 to 0.4 as $n_{\max}$ varies from 2 to 10.

Figure 3 (b) illustrates the average computational time to solve one instance for each of the different algorithms. Measurements are made on a Dual 2.4GHz Pentium IV with 2.5GB memory. The run-time for **C**+V-Ironing and **C**+V-D-Ironing also includes the time to compute payments. The performance of **C** and its ironing variants scales well with the number of arrivals $n_{\max}$ while tree-sampling was hard to scale beyond around

[6] Because allocations are for l_i periods, an agent with arrival a_i, duration δ_i and request length l_i is modeled with corresponding departure $d_i + \delta_i - l_i$ because it has no value for an allocation that is initiated in any later period.

[7] Parameters were optimized offline for each algorithm, to strike a balance between run-time and performance. Look-ahead is necessarily much smaller in tree-sampling

[8] www.ilog.com

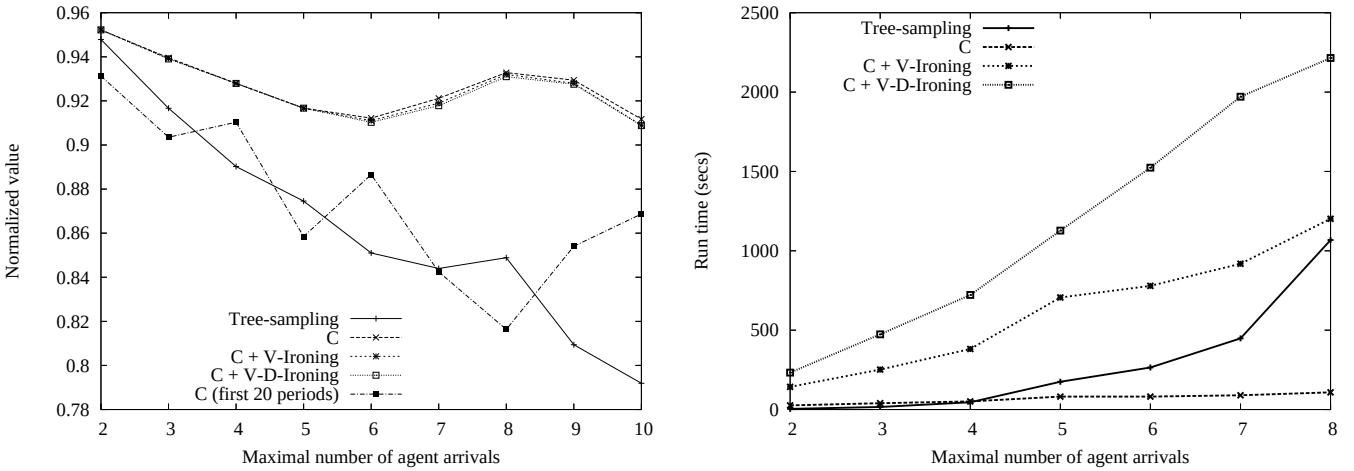

Figure 3: (a) Value of tree-sampling, **C**, **C**+V-Ironing, **C**+V-D-Ironing, and **C** evaluated on the initial 20 periods. (b) Average computational time for a single instance (including computing payments for **C**+V-Ironing and **C**+V-D-Ironing).

$n_{\max} = 8$ (even with pruning of dominated decisions.) For $n_{\max}$ greater than 6, the number of allocations proposed by **C** is fairly constant and the subsequent increase in run-time for its ironing-variants reflects the increased complexity of the offline optimization problems rather than additional monotonicity checks. The relative cost of performing ironing for $n_{\max} = 7$ is around 10x for V-Ironing and 22x for V-D-Ironing.

We also experiment with fixing q_i to 1 and/or fixing l_i to 1 for all agents. While ironing is still required with either $q_i > 1$ or $l_i > 1$, we find that **C** is already monotonic for $q_i = 1$ and $l_i = 1$ for all agents. Thus, it appears that it is the combinatorics of the domain that make **C** non-monotonic. To further explore conditions for ironing, we introduced some low-probability, high-value bids to mimic the monotonicity counterexample. Call these "shocks", and define them so that in every period, an agent with $\delta_i = 1$, $l_i = 1$, $q_i = 1$ and $r_i = 56$ arrives with some probability $p_i \in (0, 1]$. The ironing frequency increases to around 1% (5%) for **C**+V-D-Ironing and the maximal ironing occurs at $p_i \approx 0.05$ (0.02), for $S = 5$ and $S = 2$ units of supply respectively.

Conclusions

Computational ironing opens the way to leveraging general purpose online stochastic optimization within adaptive, online mechanisms. To be a practical procedure, ironing requires sensitivity information about a decision policy; e.g., so that the effect of different bid values can be traced. If a policy is already monotonic in some dimensions of the type space (e.g. with respect to arrival), then ironing only needs to be performed in the remaining dimensions. Ironing is not a panacea, however. When decisions are canceled in resource-allocation domains the resources are discarded. Future work should develop algorithms for online optimization that are already "roughly monotonic," with ironing used to make them fully monotonic at little cost in solution quality. It will also be interesting to extend these techniques to handle revenue optimality.

Acknowledgments

The authors wish to acknowledge helpful comments on this work from Florin Constantin, Rakesh Vohra and Mallesh Pai as well as the extremely useful comments from the anonymous reviewers. The first author is supported by an Alfred P. Sloan Fellowship.

References

Awerbuch, B.; Azar, Y.; and Meyerson, A. 2003. Reducing truth-telling online mechanisms to online optimization. In *Proc. ACM Symposium on Theory of Computing (STOC'03)*.

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

Bent, R., and Van Hentenryck, P. 2004. The value of Consensus in online stochastic scheduling. In *Proc. 14th Int. Conf. on Automated Planning and Scheduling*.

Conitzer, V., and Sandholm, T. 2007. Incremental mechanism design. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 1251–1256.

Hajiaghayi, M. T.; Kleinberg, R.; Mahdian, M.; and Parkes, D. C. 2005. Online auctions with re-usable goods. In *Proc. ACM Conf. on Electronic Commerce*, 165–174.

Hajiaghayi, M. T.; Kleinberg, R.; and Parkes, D. C. 2004. Adaptive limited-supply online auctions. In *Proc. ACM Conf. on Electronic Commerce*, 71–80.

Kearns, M.; Mansour, Y.; and Ng, A. 1999. A sparse sampling algorithm for near-optimal planning in large Markov Decision Processes. In *In Proc. of 16th Int. Joint Conf. on Art. Intell.*, 1324–1331.

Lavi, R., and Nisan, N. 2000. Competitive analysis of incentive compatible on-line auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*.

Myerson, R. B. 1981. Optimal auction design. *Mathematics of Operation Research* 6:58–73.

Pai, M., and Vohra, R. 2006. Notes on optimal dynamic auctions. Kellogg School of Management. Available from the authors.

Parkes, D. C., and Singh, S. 2003. An MDP-based approach to Online Mechanism Design. In *Proc. 17th Annual Conf. on Neural Inf. Proc. Systems (NIPS'03)*.

Parkes, D. C. 2007. Online mechanisms. In Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V., eds., *Algorithmic Game Theory*. Cambridge University Press. chapter 16.

Van Hentenryck, P., and Bent, R. 2006. *Online Stochastic Combinatorial Optimization*. MIT Press.