Algorithmic Game Theory

Edited by
Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay Vazirani

Contents

1

Computationally-Efficient Approximation Mechanisms

Ron Lavi

Abstract

We study the integration of game theoretic and computational considerations. Four representative subjects are discussed: (i) The monotonicity properties that decouple the game-theoretic issues from the algorithmic issues, (ii) Single-Dimensional scheduling, a problem domain that exemplifies the possibilities of monotone approximation algorithms, (iii) Incentives in Combinatorial Auctions, where the clash between CS and GT is notable and difficult, and (iv) The impossibilities of dominant-strategy implementability, where we pin-point the strong implications of this solution concept. Additional issues and open questions are briefly discussed.

1.1 Introduction

Algorithms in Computer Science, and *Mechanisms* in Game Theory, are remarkably similar objects. Both disciplines aim to implement desirable properties, drawn from "real-life" needs and limitations, but the resulting two sets of properties are completely different. A natural need is then to merge them – to simultaneously exhibit "good" game theoretic properties as well as "good" computational properties. The growing importance of the Internet as a platform for computational interactions only strengthens the motivation for this.

However, this integration task poses many difficult challenges. The two disciplines clash and contradict in several different ways, and new understandings must be obtained in order to achieve this hybridization. The classic Mechanism Design literature is rich, and contains many technical solutions when incentive issues are the key goal. Quite interestingly, most of these are not computationally efficient. In parallel, most existing algorithmic techniques, answering the computational questions at hand, do not yield the game theoretic needs. There seems to be a basic clash between classic algorithmic techniques and classic mechanism design techniques. This raises many intriguing questions: in what cases this clash is fundamental – a mathematical impossibility? Alternatively, can we "fix" this clash by applying new techniques? We will try to give a feel for these issues.

As was motivated in previous chapters, the game-theoretic quest should start with the solution concept of "implementation in dominant strategies". To design computationally-efficient dominant-strategy mechanisms, the algorithmic implications of this notion should be better understood. Section 1.2 studies this subject, describing two simple monotonicity conditions that enable the decoupling of algorithmic design from game-theoretic design. With these at hand, we can concentrate on algorithmic design, investigating the algorithmic implications of monotonicity, with no need to implicitly check for the game-theoretic properties.

As a first example to the construction of computationally-efficient approximation mechanisms we study (Section 1.3) the classic machine scheduling problem, from a game-theoretic point of view. This problem demonstrates a key difference between Economics and Computer Science – the *goals* of algorithms vs. social choice functions. While the Economics literature mainly studies welfare and/or revenue maximization, computational models raise the need for completely different objectives. In scheduling problems, a common objective is to minimize the load on the most loaded machine. As is usually the case, most existing algorithmic solutions do not yield the desired incentives, and existing techniques for incentive-compatible mechanism design do not fit such an objective. This clash was partially resolved in the recent literature, and insightful techniques have evolved.

Combinatorial Auctions (CAs) capture many of the difficult cases where the clash occurs between the incentives and computational efficiency, even though the goal is the classic economic goal of welfare maximization. Chapter 11 studies this model mostly from a computational point of view, and so our opening motivation boils down to a concrete task in this case: designing computationally efficient and incentive compatible CAs. This is the subject of Section 1.4. We first describe a technique to "plug-in" truthfulness in a certain class of algorithms, using linear programming and randomization. This is used to give tight bounds for the most general case, showing that the clash can be resolved if one is allowed to use randomness. We then attempt to avoid randomization, and in addition to answer some criticisms on direct revelation mechanisms. We describe an ascending auction for strategic players that uses a new novel solution concept. A broader view of additional mechanisms and open questions concludes this section.

In Section 1.5 we study the impossibilities associated with dominant-strategy implementability. To demonstrate positive results for Combinatorial Auctions, we have used either randomized techniques or alternative solution concepts. Does the three requirements: (i) deterministic truthfulness, (ii) computational-efficiency, and (iii) non-trivial approximation guarantees, clash in a fundamental way when one considers a "rich enough" domain like general Combinatorial Auctions? The answer to this question is, to date, unknown. But in this section we describe another, "richer" domain for which impossibility results do exist.

Let us mention two other types of GT-versus-CS clashes, not studied in this chapter, to complete the picture. *Different models:* Some CS models have a fundamentally different structure, which causes a clash even when traditional objectives are considered. In online computation, for example, players arrive over time, a fundamentally different assumption than classic mechanism design. The difficulties that emerge, and the novel solutions proposed, are discussed in Chapter 16. *Different analysis conventions:* CS usually employs worst-case analysis, avoiding strong distributional assumptions, while in Economics, the underlying distribution is usually assumed. This greatly affects the character of results, with enlightening differences. See Chapter 13 for examples.

1.2 Monotonicity and Implementability

When approaching mechanism design with traditional algorithmic tools, it would be helpful to obtain a decoupling of the game theoretic issues from the algorithmic issues. This could be achieved by formulating *algorithmic conditions* that imply the desired game-theoretic properties. If that was at hand, one could focus on algorithmic design, leaving the game-theoretic type of analysis behind. We next describe two properties, cyclic monotonicity and weak monotonicity, that achieve that.

1.2.1 A social choice setting

The section builds on the basic notions introduced in Chapter 9. As discussed there, it will be convenient to use the abstract social choice setting: there is a finite set A of alternatives, and each player has a type (valuation function) $v : A \rightarrow \Re$ that assigns a real number to every possible alternative. $v_i(a)$ should be interpreted as i's value for alternative a. The valuation function $v_i(\cdot)$ belongs to the domain V_i of all possible valuation functions.

Our goal is to implement in dominant strategies the social choice function $f : V_1 \times \cdots \times V_n \rightarrow A$. A direct implication of this implementability requirement† is the existence of a price function $p_i : V_{-i} \times A \to \Re$, for every player i, with the following property. Fix any $v_{-i} \in V_{-i}$, and any $v_i, v_i' \in V_i$. Suppose that $f(v_i, v_{-i}) = a$ and $f(v_i', v_{-i}) = b$. Then it is the case that:

$$v_i(a) - p_i(a, v_{-i}) \geq v_i(b) - p_i(b, v_{-i}) \tag{1.1}$$

In other words, player i's utility from declaring his true v_i is no less than his utility from declaring some lie, v_i', *no matter what the other players declare.* Given a social choice function f, we ask if there exist price functions that satisfy 1.1. In this section we provide conditions on f that guarantee this.

1.2.2 Two monotonicity conditions

Fix a player i, and fix the declarations of the others to v_{-i}. Let us assume, without loss of generality, that f is onto A (or, alternatively, define A' to be the range of $f(\cdot, v_{-i})$, and replace A with A' for the discussion below). Since the prices of Eq. 1.1 now become constant, we simply seek an assignment to the variables $\{p_a\}_{a \in A}$ such that $v_i(a) - v_i(b) \geq p_a - p_b$ for every $a, b \in A$ and $v_i \in V_i$ with $f(v_i, v_{-i}) = a$. This motivates the following definition:

$$\delta_{a,b} \doteq \inf\{v_i(a) - v_i(b) \mid v_i \in V_i, f(v_i, v_{-i}) = a\} \tag{1.2}$$

With this we can rephrase the above assignment problem, as follows. We seek an assignment to the variables $\{p_a\}_{a \in A}$ that satisfies:

$$p_a - p_b \leq \delta_{a,b} \quad \forall a, b \in A \tag{1.3}$$

This is easily solved by looking at the *representation graph*:

Definition 1.1 The representation graph of a social choice function f is a directed weighted graph $G = (V, E)$ where $V = A$ and $E = A \times A$. The weight $w_{a,b}$ of an edge $a \to b$ (for any $a, b \in A$) is $\delta_{a,b}$.

In other words, the graph nodes are exactly the alternatives of A, and for any $a, b \in A$ we have a directed edge $a \to b$ with weight $w_{a,b} = \delta_{a,b}$. We can now apply a standard basic result of graph theory:

Proposition 1.2 *There exists a feasible assignment to 1.3 if and only if the representation graph has no negative-length cycles. Furthermore, if all cycles are non-negative, the feasible assignment is as follows: set p_a to the length of the shortest path from a to some arbitrary fixed node $a^* \in A$.*

† See Chapter 9 for full details.

Proof Suppose first that there exists a negative-length cycle $a_1, a_2, ..., a_k$, i.e. $a_1 = a_k$ and $\sum_{i=1}^{k} \delta_{a_i,a_{i+1}} < 0$. Consider the following inequalities:

$$\begin{gathered} p_{a_1} - p_{a_2} \leq \delta_{a_1,a_2} \\ \vdots \\ p_{a_{k-1}} - p_{a_k} \leq \delta_{a_{k-1},a_k} \end{gathered}$$

Summing these $k-1$ inequalities, the left-hand side sums to zero, while the right-hand side is the cycle's length, which is negative. Thus no assignment of $p_{a_1}, ..., p_{a_{k-1}}$ satisfies all these inequalities. This shows one direction. Now suppose that every cycle is non-negative, fix some arbitrary $a^* \in A$, and set p_a to the length of the shortest path from a to a^* (since there are no negative cycles, the shortest path is well defined). Let us verify that $p_a - p_b \leq \delta_{a,b}$ for any $a, b \in A$. The shortest path from a to $a*$ is not longer than $w_{a,b}$ plus the shortest path b to a^*. I.e. $p_a \leq \delta_{a,b} + p_b$. □

With this we can easily state a condition for implementability:

Definition 1.3 (Cyclic monotonicity) A social choice function f satisfies cyclic monotonicity if for every player i, $v_{-i} \in V_{-i}$, some integer $k \leq |A|$, and $v_i^1, ..., v_i^k \in V_i$,

$$\sum_{j=1}^{k} [v_i^j(a_j) - v_i^j(a_{j+1})] \geq 0$$

where $a_j = f(v_i^j, v_{-i})$ for $1 \leq j \leq k$, and $a_{k+1} = a_1$.

Proposition 1.4 *f satisfies cyclic monotonicity if and only if the representation graph of f has no negative cycles.*

Proof If the graph has no negative cycles then $\sum_{j=1}^{k}[v_i^j(a_j) - v_i^j(a_{j+1})] \geq \sum_{j=1}^{k} \delta_{a_j,a_{j+1}} \geq 0$. In the other direction, suppose that the cyclic monotonicity condition is violated with $v_i^1, ..., v_i^k$. Therefore $\sum_{j=1}^{k} \delta_{a_j,a_{j+1}} \leq \sum_{j=1}^{k}[v_i^j(a_j) - v_i^j(a_{j+1})] < 0$ and so the cycle $a_1, ..., a_k, a_1$ is negative. □

Corollary 1.5 *A social choice function f is dominant-strategy implementable if and only if it satisfies cyclic monotonicity.*

Cyclic monotonicity satisfies our motivating goal: a condition on f that involves only the properties of f, without existential price qualifiers. However, it is quite complex. k could be large, and a "shorter" condition would have been nicer. "Weak monotonicity" (W-MON) is exactly that:

Definition 1.6 (Weak monotonicity) A social choice function f satisfies W-MON if for every player i, every v_{-i}, and every $v_i, v_i' \in V_i$ with $f(v_i, v_{-i}) = a$ and $f(v_i', v_{-i}) = b$, $v_i'(b) - v_i(b) \geq v_i'(a) - v_i(a)$, In other words, if the outcome changes from a to b when i changes her type from v_i to v_i' then i's value for b has increased at least as i's value for a in the transition v_i to v_i'. Note that W-MON is equivalent to cyclic monotonicity with $k = 2$, or, alternatively, to the requirement that the representation graph has no negative *two-edge* cycles. Hence it is necessary for truthfulness. As it turns out, many times it is also sufficient:

Theorem 1.7 *If the domain V_i is convex then any social choice function f that satisfies W-MON is dominant-strategy implementable.*

We next give a proof for the special case of "order-based" domains. Recall that we fix a player i and some $v_{-i} \in V_{-i}$. Since we are interested only in i, w.l.o.g. for any $a, b \in A$ there exists $v_i \in V_i$ such that $v_i(a) \neq v_i(b)$, i.e. a and b are not inherently the same (otherwise we remove b from A).

Definition 1.8 (Order-based domain) A domain V_i is "order-based" if there exists a partial order $\succeq_i$ over the set A such that for any $v_i \in \Re^{|A|}$, $v_i \in V_i$ if and only if $v_i(a) \geq v_i(b)$ for every $a, b \in A$ with $a \succeq_i b$.

Note that the "if and only if" statement implies that *every* $v_i \in \Re^{|A|}$ that "respects" $\succeq_i$ must belong to V_i. Thus for example V_i is unbounded. CAs (see Chapter 11) are an example for an ordered-based domain. An alternative is simply a subset of items (since we are in a single player setting), and we naturally get the partial order $\succeq_i$ since for any $S \subset T$, $v_i(S) \leq v_i(T)$.†

Theorem 1.9 *If the domain V_i is ordered-based then any social choice function f that satisfies W-MON is dominant-strategy implementable.*

Proof We need several claims to prove the theorem.

Claim 1.10 *If $f(v_i) = a$ then $v_i(x) - v_i(a) \leq \delta_{xa}$ for any $x \in A$.*

Proof W-MON implies that every two-edge cycle is non-negative, i.e. $\delta_{xa} + \delta_{ax} \geq 0$. By definition, we get $v_i(a) - v_i(x) \geq \delta_{ax} \geq -\delta_{xa}$. □

Claim 1.11 *If $x \succeq_i a$ then for any $c \in A$, $\delta_{cx} \leq \delta ax$.*

Proof $\forall v_i \in V_i$, $v_i(a) \leq v_i(x)$, hence $v_i(c) - v_i(x) \leq v_i(c) - v_i(a)$. □

Let $c \in A$ be some maximal alternative w.r.t. $\succeq_i$, i.e. for any $a \neq c$, $a \not\succeq_i c$. Thus, for any $v_i \in V_i$ and any $\Delta > 0$, the valuation v_i' that is defined to be $v_i'(a) = v_i(a)$ for $a \neq c$ and $v_i'(c) = v_i(c) + \Delta$ must belong to V_i.

Claim 1.12 *For any $a, b \in A$, $\delta_{ab} \geq \delta_{cb} - \delta_{ca}$.*

† However, special cases of CAs, e.g. bidders with submodular valuations, are not ordered based since a partial order by itself cannot capture the additional structure imposed there.

Proof It suffices to show that for every v_i with $f(v_i) = a$, $v_i(a) - v_i(b) \geq \delta_{cb} - \delta_{ca}$. Suppose by contradiction that $v_i(a) - v_i(b) < \delta_{cb} - \delta_{ca}$. Rearranging, and using Claim 1.10 with $x = c$, we get that $v_i(c) \leq v_i(a) + \delta_{ca} < v_i(b) + \delta_{cb}$. Let $X = \{x \in A \mid x \succeq_i a \text{ and } v_i(x) = v_i(a)\}$. If we wish to increase a's value by ϵ, we need to increase the value of every alternatives in X by ϵ, to maintain the order $\succeq_i$. Fix some $\epsilon > 0$ and $\Delta > 0$ such that $v_i(a) + \epsilon + \delta_{ca} < v_i(c) + \Delta < v_i(b) + \delta_{cb}$, and let v_i' be:

$$v_i'(x) = \begin{cases} v_i(x) + \Delta & x = c \\ v_i(x) + \epsilon & x \in X \cup \{a\} \setminus \{c\} \\ v_i(x) & \text{otherwise} \end{cases}$$

By the choice of c and X, v_i' respects $\succeq_i$ and therefore $v_i' \in V_i$. Since $v_i'(a) - v_i(a) = \epsilon > 0$, W-MON implies that $f(v_i') \in X \cup \{a, c\}$. For every $x \in X \cup \{a\} \setminus \{c\}$, we have that $v_i'(x) = v_i'(a)$ (by construction of X and v_i') and $\delta_{cx} \leq \delta_{ca}$ (by claim 1.11). Thus $v_i'(x) + \delta_{cx} \leq v_i'(a) + \delta_{ca} < v_i'(c)$. Rearranging, we get $v_i'(x) - v_i'(c) < -\delta_{cx} \leq \delta_{xc}$, implying that $f(v_i') \neq x$. Thus $f(v_i') = c$. But $v_i'(c) - v_i'(b) < \delta_{cb}$, a contradiction. □

By Claim 1.12, we get an assignment that satisfies 1.3: For any $x \in A$ set $p_x = -\delta_{cx}$. This concludes the proof of the theorem. □

W-MON is sufficient for implementability for every convex domain, but this does not hold for every domain. An exact characterization is missing:

Open Question Exactly characterize the domains for which W-MON is sufficient for implementability.

1.3 Single-Dimensional Domains and Job Scheduling

As a first example for the interaction between game theory and algorithmic theory, we consider single-dimensional domains. Simple single-dimensional domains were introduced in Chapter 9, where every alternative is either a winning or a losing alternative for each player. Here we discuss a more general case. Intuitively, single-dimensionality implies that a single parameter determines the player's valuation vector. In chapter 9, this was simply the value for winning, but less straight-forward cases also exist:

Scheduling related machines. In this domain, n jobs are to be assigned to m machines, where job j consumes p_j time-units, and machine i has speed s_i. Thus machine i requires p_j/s_i time-units to complete job j. Let $l_i = \sum_{j|\ j \text{ is assigned to } i} p_j$ be the load on machine i. Our schedule aims to minimizes the term $\max_i l_i/s_i$, (the *makespan*). Each machine is a

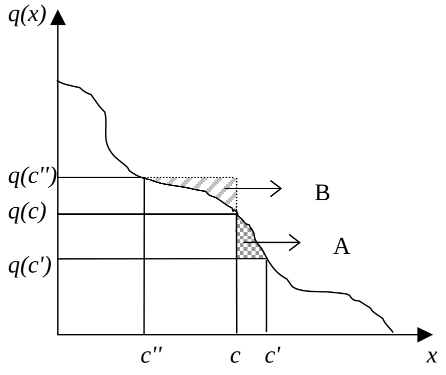

Fig. 1.1. A monotone load curve.

selfish entity, incuring a constant cost for every consumed time unit (and w.l.o.g. assume this cost is 1). Thus the utility of a machine from a load l_i and a payment P_i is $-l_i/s_i - P_i$. The mechanism designer knows the processing times of the jobs, and constructs a scheduling mechanism.

Although here the set of alternatives cannot be partitioned to “wins” and “loses”, this is clearly a single-dimensional domain:

Definition 1.13 (single-dimensional linear domains) A domain V_i of player i is single-dimensional and linear if there exist non-negative real constants (the “loads”) $\{q_{i,a}\}_{a \in A}$ such that, for any $v_i \in V_i$, there exists $c \in \Re_-$ (the “cost”) such that $v_i(a) = q_{i,a} \cdot c$.

In other words, the type of a player is simply her cost c – disclosing it gives us the entire valuation vector. Note that the scheduling domain is indeed single-dimensional and linear: the parameter c is equal to $1/s_i$, and the constant $q_{i,a}$ for alternative a is the load assigned to i according to a.

A natural symmetric definition exists for value-maximization (as opposed to cost-minimization) problems, where the types are non-negative.

We aim to design a computationally-efficient approximation algorithm, *that is also implementable.* As the social goal is a certain min-max criteria, and not to minimize the *sum* of costs, we cannot use the general VCG technique. Since we have a convex domain, we already know that we need a weakl monotone algorithm. But what exactly does this mean? Luckily, the formulation of weak monotonicity can be much simplified for this case.

If we fix the costs c_{-i} declared by the other players, an algorithm for a single-dimensional linear domain determines the load $q_i(c)$ of player i as a function of her reported cost c. Take two possible types c and c', and suppose $c' > c$. Then weak monotonicity reduces to $-q_i(c')(c'-c) \geq -q_i(c)(c'-c)$, which holds iff $q_i(c') \leq q_i(c)$. Hence by theorem 1.7 we get that such an algorithm is implementable if and only if its load functions are monotone non-increasing. Figure 1.1 describes this, and will help us to figure out the required prices for implementability.

Suppose that we charge a payment of $P_i(c) = \int_0^c [q_i(x) - q_i(c)]dx$ from player i if he declares a cost of c. Using Fig. 1.1, we can easily verify that these prices lead to incentive compatibility: Suppose player i's true cost is c. If he reports the truth his utility is the entire area below the load curve up to c. Now if he declares some $c' > c$, his utility will decrease by exactly the area marked by A: his cost from the resulting load will indeed decrease to $c \cdot q_i(c')$, but his payment will increase to be the area between the line $q_i(c')$ and the load curve. On the other hand, if the player will report $c'' < c$, his utility will decrease by exactly the area marked by B, since his cost from the resulting load will increase to $c \cdot q_i(c'')$. Thus these prices satisfy the incentive-compatibility inequalities, and in fact this is a simple direct proof for the sufficiency of load-monotonicity for this case.

The above prices do not satisfy individual rationality, since a player always incurs a negative utility if we use these prices. To overcome this, the usual exercise is to add a large enough constant to the prices, which in our case can be $\int_0^\infty q_i(x)dx$. Note that if we add this to the above prices we get that a player that does not receive any load (i.e. declares a cost of infinity) will have a zero utility, and in general the utility of a truthful player will be non-negative, exactly $\int_c^\infty q_i(x)dx$. From all the above we get:

Theorem 1.14 *An algorithm for a single-dimensional linear domain is implementable if and only if its load functions are non-increasing. Furthermore, if this is the case then charging from every player i a price*

$$P_i(c) = \int_0^c [q_i(x) - q_i(c)]dx - \int_c^\infty q_i(x)dx$$

will result in an individually rational dominant strategy implementation.

In the application to scheduling, we will use a randomized mechanism, and will employ truthfulness in expectation (see Chapter 9, Definition 9.27). One should observe that, from the discussion above, it follows that truthfulness in expectation is equivalent to the monotonicity of the *expected* load.

1.3.1 A monotone algorithm for the job scheduling problem

Now that we understand the exact form of an implementable algorithm, we can construct one that achieves a reasonable approximation. The first attempt could be to check if exisiting approximation algorithms for the problem are implementable, but the answer is unfortunately no. We next show

how this clash can be partially resolved by designing a monotone approximation algorithm. Note that we face a "classic" algorithmic problem – no game-theoretic issues are left for us to handle.

Before we start, we assume that jobs and machines are reordered so that $s_1 \geq s_2 \geq \cdots \geq s_m$ and $p_1 \geq p_2 \geq \cdots \geq p_n$. For the algorithmic construction, we first need to estimate the optimal makespan of a given instance.

Estimating the optimal makespan. Fix a job-index j, and some target makespan T. If a schedule has makespan at most T then it must assign any job out of $1, ..., j$ to a machine i such that $T \geq p_j/s_i$. Let $i(j,T) = max\{i \mid T \geq p_j/s_i \}$. Thus any schedule with makespan at most T assigns jobs $1, ..., j$ to machines $1, ..., i(j,T)$. From space considerations, it immediately follows that:

$$T \geq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i(j,T)} s_l}. \tag{1.4}$$

Now define

$$T_j = \min_i \max\{\frac{p_j}{s_i}, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} s_l}\} \tag{1.5}$$

Lemma 1.15 *For any job-index j, the optimal makespan is at least T_j.*

Proof Fix any $T < T_j$. We prove that T violates 1.4, hence cannot be any feasible makespan, and the claim follows. Let i_j be the index that determines T_j. The left expression in the max term is increasing with i, while the right term is decreasing. Thus i_j is either the last i where the right term is larger than the left one, or the first i for which the left term is larger than the right one. We prove that T violates 1.4 for each case seperately.

Case 1 ($\frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j} s_l} \geq \frac{p_j}{s_{i_j}}$): For $i_j + 1$ the max term is received by $\frac{p_j}{s_{i_j+1}}$, Since T_j is the min-max, we get $T_j \leq \frac{p_j}{s_{i_j+1}}$. Since $T < T_j$, we have $i(j,T) \leq i_j$, and $T < T_j = \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j} s_l} \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i(j,T)} s_l}$. Hence T violates 1.4, as claimed.

Case 2 ($\frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j} s_l} < \frac{p_j}{s_{i_j}}$): $T_j \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j-1} s_l}$ since T_j is the min-max, and the max for $i_j - 1$ is received at the right. Additionally, $i(j,T) < i_j$ since $T_j = \frac{p_j}{s_{i_j}}$ and $T < T_j$. Thus $T < T_j \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j-1} s_l} \leq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i(j,T)} s_l}$, as we need. □

With this, we get a good lower bound estimate of the optimal makespan:

$$T_{LB} = \max_j T_j \tag{1.6}$$

The optimal makespan is at least T_j for any j, hence it is at least T_{LB}.

The algorithm. We start with a fractional schedule. If machine i gets an α fraction of job j then the resulting load is assumed to be $(\alpha \cdot p_j)/s_i$. This is of-course not a valid schedule, and we later round it to an integral one.

Definition 1.16 (The fractional allocation) Let j be the first job such that $\sum_{k=1}^{j} p_k > T_{LB} \cdot s_1$. Assign to machine 1 jobs $1, ..., j-1$, plus a fraction of j in order to equate $l_1 = T_{LB} \cdot s_1$. Continue recursively with the unassigned fractions of jobs and with machines $2, ..., m$.

Lemma 1.17 *There is enough space to fractionally assign all jobs, and if job j is fractionally assigned to machine i then $p_j/s_i \le T_{LB}$.*

Proof Let i_j be the index that determines T_j. Since $T_{LB} \ge T_j \ge \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i_j} s_l}$, we can fractionally assign jobs $1, .., j$ up to machine i_j. Since $T_j \ge p_j/s_{i_j}$ we get the second part of the claim, and setting $j = n$ gives the first part. □

Lemma 1.18 *The fractional load function is monotone.*

Proof We show that if s_i increases to $s'_i = \alpha \cdot s_i$ (for $\alpha > 1$) then $l'_i \le l_i$. Let T'_{LB} denote the new estimate of the optimal makespan. We first claim that $T'_{LB} \le \alpha \cdot T_{LB}$. For an instance $s''_1, ..., s''_m$ such that $s''_l = \alpha \cdot s_l$ for all machines l we have that $T''_{LB} = \alpha \cdot T_{LB}$ since both terms in the max expression of T_j were multiplied by α. Since $s'_l \le s_l$ for all l we have that $T'_{LB} \le T''_{LB}$. Now, if $l_i = T_{LB} \cdot s_i$, i.e. i was full, then $l'_i \le T'_{LB} \cdot s'_i \le T_{LB} \cdot s_i = l_i$. Otherwise $l_i < T_{LB} \cdot s_i$, hence i is the last non-empty machine. Since $T'_{LB} \ge T_{LB}$, all previous machines now get at least the same load as before, hence machine i cannot get more load. □

We now round to an integral schedule. The natural rounding, of integrally placing each job on one of the machines that got some fraction of it, provides a 2-approximation, but violates the required monotonicity (see the exercises). Thus we resort to randomization:

Definition 1.19 (Rounding the schedule) Choose $\alpha \in [0, 1]$ uniformly at random. For every job j that was fractionally assigned to i and $i+1$, if j's fraction on i is at least α, assign j to i in full, otherwise assign j to $i+1$.

Theorem 1.20 *The randomized scheduling algorithm is truthful in expectation, and obtains a 2-approx. to the optimal makespan in polynomial-time.*

Proof Let us check the approximation first. A machine i may get, in addition to its full jobs, two more jobs. One, j, is shared with machine $i-1$, and the other, k, is shared with machine $i+1$. If j was rounded to i then i initially has at least $1-\alpha$ fraction of j, hence the additional load caused by j is at most $\alpha \cdot p_j$. Similarly, If k was rounded to i then i initially has at least α

fraction of k, hence the additional load caused by k is at most $(1-\alpha)\cdot p_k$. Thus the maximal total additional load that i gets is $\alpha \cdot p_j + (1-\alpha)\cdot p_k$. By Lemma 1.17 we have that $\max\{p_j, p_k\} \leq T_{LB}$ and since T_{LB} is not larger than the otpimal maximal makespan, the approximation claim follows.

For truthfulness, we only need that the expected load is monotone. Note that machine $i-1$ gets job j with probablity α, so i gets it with probability $1-\alpha$, and i gets k with probability α. So the expected load of machine i is exactly its fractional load. The claim now follows from Lemma 1.18. □

A note about price computation is in place. A polynomial-time *mechanism* must compute the prices in polynomial time. To compute the prices for our mechanism we need to integrate over the load function of a player, fixing the others' speeds. This is a step function, with polynomial number of steps (when a player declares a large enough speed she will get all jobs, and as she decreases her speed more and more jobs will be assigned elsewhere, where the set of assigned jobs will decrease monotonically). Thus we have verified that price computation is polynomial-time in our case.

1.4 Incentives in Combinatorial Auctions

Combinatorial Auctions (CAs) are a central model with theoretical importance and practical relevancy. It generalizes many theoretical algorithmic settings, like job scheduling and network routing, and is evident in many real-life situations. Chapter 11 is exclusively devoted to Combinatorial Auctions, providing a comprehensive discussion on the model and its various computational aspects. Here, we touch on a different, important point: how to design CAs that are, simultaneously, computationally efficient *and* incentive-compatible. While each aspect is important on its own, obviously only the integration of the two provides an acceptable solution.

Let us shortly restate the essentials. In a CA, we allocate m items (Ω) to n players. Players value *subsets* of items, and $v_i(S)$ denotes i's value of a bundle $S \subseteq \Omega$. Valuations additionally satisfy: (i) monotonicity, i.e $v_i(S) \leq v_i(T)$ for $S \subseteq T$, and (ii) normalization, i.e. $v_i(\emptyset) = 0$. In this section we consider the goal of maximizing the social welfare: find an allocation $(S_1, ..., S_n)$ that maximizes $\sum_i v_i(S_i)$.

Since a general valuation has size exponential in$n$ and m, the representation issue must be discussed. Chapter 11 examines two models. In the *bidding languages* model, the bid of a player represents his valuation is a concise way. For this model it is NP-hard to obtain an approximation of $\Omega(m^{1/2-\epsilon})$, for any $\epsilon > 0$ (if single-minded bids are allowed). In the *query* *access* model, the mechanism iteratively queries the players in the course of computation. For this model, any algorithm with polynomial communication cannot obtain an approximation of $\Omega(m^{1/2-\epsilon})$ for any $\epsilon > 0$. These bounds are tight, as there exist a deterministic $\sqrt{m}$-approximation with polynomial computation and communication. Thus, for the general case, the computational status by itself is well-understood.

The basic incentives issue is again well-understood: with VCG (which requires the exact optimum) we can obtain truthfulness. The two considerations therefore clash. We next detail two mechanisms that resolve this. The first is a randomized mechanism for the general case, which develops an LP-based technique to convert algorithms to truthful mechanisms. The second is a deterministic mechanism for a special case of single-value players, that takes the useful form of an ascending auction, and uses a novel solution concept to analyze the strategic properties. A short survey of other mechanisms and open problems concludes the section.

1.4.1 A randomized mechanism for the general case

We describe a rather general technique to convert approximation algorithms to truthful mechanisms, by using randomization: given any algorithm to the general CA problem that outputs a c-approximation to the optimal *fractional* social welfare, one can construct a randomized c-approximation mechanism that is truthful in expectation. Thus, the same approximation guarantee is maintained. The construction and proof are described in three steps. We first discuss the fractional domain, where we allocate fractions of items. We then show how to move back to the original domain while maintaining truthfulness, by using randomization. This uses an interesting decomposition technique, which we then describe.

The fractional domain. Let $x_{i,S}$ denote the fraction of subset S that player i receives in allocation x. Assume that her value for that fraction is $x_{i,S} \cdot v_i(S)$. The welfare-maximization becomes an LP:

$$\max \quad \sum_{i,S\neq\emptyset} x_{i,S}\cdot v_i(S) \tag{CA-P}$$

$$\text{subject to} \quad \sum_{S\neq\emptyset} x_{i,S} \leq 1 \qquad \text{for each player } i \tag{1.7}$$

$$\sum_i \sum_{S:j\in S} x_{i,S} \leq 1 \qquad \text{for each item } j \tag{1.8}$$

$$x_{i,S} \geq 0 \qquad \forall i, S \neq \emptyset.$$

By constraint 1.7, a player receives at most one integral subset, and constraint 1.8 ensures that each item is not over-allocated. The empty set is excluded for technical reasons that will become clear below. This LP is solvable in time polynomial in its size by using e.g. the ellipsoid method. Its size is related to our representation assumption. If we assume the bidding languages model, where the LP has size polynomial in the size of the bid (for example k-minded players), then we have a polynomial-time algorithm. If we assume general valuations and a query-access, this LP is solvable with a polynomial number of demand queries (see Chapter 11). Note that, in either case, the number of non-zero $x_{i,S}$ coordinates is polynomial, since we obtain x in polynomial-time (this will become important below). In addition, since we obtain the *optimal allocation*, we can use VCG (see Chapter 9) to get:

Proposition 1.21 *In the fractional case, there exists a truthful optimal mechanism with efficient computation and communication, for both the bidding languages model and the query-access model.*

The transition to the integral case. The following technical lemma allows for an elegant transition, by using randomization.

Definition 1.22 Algorithm A "verifies a c-integrality-gap" (for the linear program CA-P) if it receives as input real numbers $w_{i,S}$, and outputs an integral point $\tilde{x}$ which is feasible for CA-P, and

$$c \cdot \sum_{i,S} w_{i,S} \cdot \tilde{x}_{i,S} \geq \max_{\text{feasible x's}} \sum_{i,S} w_{i,S} \cdot x_{i,S}$$

Lemma 1.23 (The decomposition lemma) *Suppose A verifies a c-integrality-gap for CA-P (in polynomial time), and x is any feasible point of CA-P. Then one can decompose x/c to a convex combination of integral feasible points. Furthermore, this can be done in polynomial-time.*

Let $\{x^l\}_{l\in\mathcal{I}}$ be all integral allocations. The proof will find $\{\lambda_l\}_{l\in\mathcal{I}}$ such that (i) $\forall l \in \mathcal{I},\ \lambda_l \geq 0$, (ii) $\sum_{l\in\mathcal{I}} \lambda_l = 1$, and (iii) $\sum_{l\in\mathcal{I}} \lambda_l \cdot x^l = x/c$. We will also need to provide the integrality gap verifier. But first we show how to use all this to move back to the integral case, while maintaining truthfulness.

Definition 1.24 (The decomposition-based mechanism)

(i) Compute an optimal fractional solution, x^*, and VCG prices $p_i^F(v)$.
(ii) Obtain a decomposition $x^*/c = \sum_{l\in\mathcal{I}} \lambda_l \cdot x^l$.
(iii) With probability λ_l: (i) choose allocation x^l, (ii) set prices $p_i^R(v) = [v_i(x^l)/v_i(x^*)]p_i^F(v)$.

The strategic properties of this mechanism hold whenever the *expected* price equals the fractional price over c. The specific prices chosen satisfy, in addition to that, strong individual rationality (i.e truth-telling ensures a non-negative utility, regardless of the randomized choice): VCG is individually rational, hence $p_i^F(v) \leq v_i(x^*)$. Thus $p_i^R(v) \leq v_i(x^l)$ for any $l \in \mathcal{I}$.

Lemma 1.25 *The decomposition-based mechanism is truthful in expectation, and obtains a c-approximation to the social welfare.*

Proof The expected social welfare of the mechanism is $(1/c)\sum_i v_i(x^*)$, and since x^* is the optimal fractional allocation, the approximation guarantee follows. For truthfulness, we first need that the expected price of a player equals her fractional price over c, i.e. $E_{\lambda_l}[p_i^R(v)] = p_i^F(v)/c$:

$$E_{\{\lambda_l\}_{l\in\mathcal{I}}}[p_i^R(v)] = \textstyle\sum_{l\in\mathcal{I}} \lambda_l \cdot [v_i(x^l)/v_i(x^*)] \cdot p_i^F(v) = \qquad (1.9)$$
$$[p_i^F(v)/v_i(x^*)] \cdot \textstyle\sum_{l\in\mathcal{I}} \lambda_l \cdot v_i(x^l) = [p_i^F(v)/v_i(x^*)] \cdot v_i(x^*/c) = p_i^F(v)/c$$

Fix any $v_{-i} \in V_{-i}$. Suppose that when i declares v_i, the fractional optimum is x^*, and when she declares v_i', the fractional optimum is z^*. The VCG fractional prices are truthful, hence

$$v_i(x^*) - p_i^F(v_i, v_{-i}) \geq v_i(z^*) - p_i^F(v_i', v_{-i}) \qquad (1.10)$$

By 1.9 and by the decomposition, dividing 1.10 by c yields

$$[\sum_{l\in\mathcal{I}} \lambda_l \cdot v_i(x^{*^l})] - E_{\lambda_l}[p_i^R(v_i, v_{-i})] \geq [\sum_{l\in\mathcal{I}} \lambda_l \cdot v_i(z^{*^l})] - E_{\lambda_l}[p_i^R(v_i', v_{-i})]$$

The left hand side is the expected utility for declaring v_i and the right hand side is the expected utility for declaring v_i', and the lemma follows. □

The above analysis is for one-shot mechanisms, where a player declares his valuation up-front (the bidding languages model). For the query-access model, where players are being queried *iteratively*, the above analysis leads to a weakening of the solution concept to *ex-post Nash*: if all other players are truthful, player i will maximize his expected utility by being truthful.

For example, consider the following single item auction for two players: player I bids first, player II observes I's bid *and then* bids. The highest bidder wins and pays the second highest value. Here, truthfulness fails to be a dominant: Suppose II chooses the strategy "if I bids above 5, I bid 20, otherwise I bid 2". If I's true value is 6, his best response is to declare 5. However, truthfulness is an ex-post Nash equilibrium: if II fixes *any* value and bids that, then, regardless of II's bid, I's best response is the truth.

In our case, if all others answer queries truthfully, the analysis carry through as is, and so truth-telling maximizes i's the expected utility. The decomposition-based mechanism thus has truthfulness-in-expectation as an ex-post Nash equilibrium for the query-access model. Putting it differently, even if a player was told beforehand the types of the other players, he would have no incentive to deviate from truth-telling.

The decomposition technique. We now decompose $x/c = \sum_{l\in\mathcal{I}} \lambda_l \cdot x^l$, for any x feasible to CA-P. We first write the LP P and its dual D. Let $E = \{(i,S)|x_{i,S} > 0\}$. Recall that E is of polynomial size.

$$\begin{array}{ll} \min \sum_{l\in\mathcal{I}} \lambda_l \quad \text{(P)} & \max \frac{1}{c} \sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \quad \text{(D)} \\ \text{s.t.} & \text{s.t.} \\ \sum_l \lambda_l x^l_{i,S} = \frac{x_{i,S}}{c} \quad \forall (i,S)\in E \quad (1.11) & \sum_{(i,S)\in E} x^l_{i,S} w_{i,S} + z \le 1 \ \forall l \in \mathcal{I} \quad (1.12) \\ \sum_l \lambda_l \ge 1 & z \ge 0 \\ \lambda_l \ge 0 \quad \forall l \in \mathcal{I} & w_{i,S} \text{ unconstrained} \quad \forall (i,S)\in E. \end{array}$$

Constraints 1.11 of P describe the decomposition, hence if the optimum satisfies $\sum_{l\in I} \lambda_l = 1$ we are almost done. P has exponentially many variables, so we need to show how to solve it in polynomial time. The dual D will help. It has variables $w_{i,S}$ for each constraint 1.11 of P, so it has polynomially many variables but exponentially many constraints. We use the ellipsoid method to solve it, and construct a separation oracle using our verifier A.

Claim 1.26 *If w, z is feasible for D then $\frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \le 1$. Furthermore, if this inequality is reversed, one can use A to find a violated constraint of D in polynomial-time.*

Proof Suppose $\frac{1}{c} \cdot \sum_{(i,S)\in E} x_{i,S} w_{i,S} + z > 1$. Let A receive w as input and suppose that the integral allocation that A outputs is x^l. We have $\sum_{(i,S)\in E} x^l_{i,S} w_{i,S} \ge \frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} > 1 - z$, where the first inequality follows since A is a c-approximation to the fractional optimum, and the second inequality is the violated inequality of the claim. Thus constraint 1.12 is violated (for x^l). □

Corollary 1.27 *The optimum of D is 1, and the decomposition $x/c = \sum_{l\in\mathcal{I}} \lambda_l \cdot x^l$ is polynomial-time computable.*

Proof $z = 1, w_{i,S} = 0 \ \forall (i,S) \in E$ is feasible, hence the optimum is at least 1. By claim 1.26 it is at most 1. To solve P, we first solve D with the following separation oracle: given w, z, if $\frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \le 1$, return the separating hyperplane $\frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z = 1$. Otherwise, find the violated constraint, which implies the separating hyperplane. The ellipsoid method uses polynomial number of constraints, thus there is an equivalent program with only those constraints. Its dual is a program that is equivalent to P but with polynomial number of variables. We solve that to get the decomposition. $\square$

Verifying the integrality gap. We now construct the integrality gap verifier for CA-P. Recall that it receives as input weights $w_{i,S}$, and outputs an integral allocation x^l which is a c-approximation to the social welfare w.r.t. $w_{i,S}$. Two requirements differentiate it from a "regular" c-approximation for CAs: (i) it cannot assume any structure on the weights $w_{i,S}$ (unlike CA, where we have non-negativity and monotonicity), and (ii) the obtained welfare must be compared to the *fractional* optimum (usually we care for the integral optimum). The first property is not a problem:

Claim 1.28 *Given a c-approximation for general CAs, A', where the approximation is with respect to the fractional optimum, one can obtain an algorithm A that verifies a c-integrality-gap for the linear program CA-P, with a polynomial time overhead on top of A.*

Proof Given $w = \{w_{i,S}\}_{(i,S)\in E}$, define w^+ by $w^+_{i,S} = \max(w_{i,S}, 0)$, and $\tilde{w}$ by $\tilde{w}_{i,S} = \max_{T\subseteq S\,,\,(i,T)\in E} w^+_{i,T}$ (where the maximum is 0 if no $T \subseteq S$ has $(i,T) \in E$. $\tilde{w}$ is a valid valuation, and can be succinctly represented with size $|E|$. Let $O^* = \max_{x \text{ is feasible for CA-P}} \sum_{(i,S)\in E} x_{i,S} w_{i,S}$. Feed $\tilde{w}$ to A' to get $\tilde{x}$ such that $\sum_{i,S} \tilde{x}_{i,S}\tilde{w}_{i,S} \geq \frac{O^*}{c}$ (since $\tilde{w}_{i,S} \geq w_{i,S}$ for every (i,S)).

Note that it is possible that $\sum_{(i,S)\in E} \tilde{x}_{i,S} w_{i,S} < \sum_{i,S} \tilde{x}_{i,S}\tilde{w}_{i,S}$, since (i) the left hand sum only considers coordinates in E and (ii) some $w_{i,S}$ coordinates might be negative. To fix the first problem define x^+ as follows: for any (i,S) such that $\tilde{x}_{i,S} = 1$, set $x^+_{i,T'} = 1$ for $T' = \arg\max_{T\subseteq S:(i,T)\in E} w^+_{i,T}$ (set all other coordinates of x^+ to 0). By construction, $\sum_{i,S} \tilde{x}_{i,S}\tilde{w}_{i,S} = \sum_{(i,S)\in E} x^+_{i,S} w^+_{i,S}$. To fix the second problem define x^l as follows: set $x^l_{i,S} = x^+_{i,S}$ if $w_{i,S} \geq 0$ and 0 otherwise. Clearly, $\sum_{(i,S)\in E} x^l_{i,S} w_{i,S} = \sum_{(i,S)\in E} x^+_{i,S} w^+_{i,S}$, and x^l is feasible for CA-P. $\square$

The requirement to approximate the fractional optimum does affect generality. However, one can use the many algorithms that use the primal-dual method, or a derandomization of an LP randomized rounding. Simple combinatorial algorithms may also satisfy this property. In fact, the greedy algorithm from Chapter 11 for single-minded players satisfies the requirement, and a natural variant verifies a $\sqrt{2}\cdot\sqrt{m}$ integrality-gap for CA-P:

Definition 1.29 (Greedy (revisited)) Fix $\{w_{i,S}\}_{(i,S)\in E}$ as the input. Construct x as follows. Let $(i,S) = \arg\max_{(i',S')\in E}(w_{i',S'}/\sqrt{|S'|})$. Set $x_{i,S} = 1$. Remove from E all (i',S') with $i' = i$ or $S' \cap S \neq \emptyset$. If $E \neq \emptyset$, reiterate.

Lemma 1.30 *Greedy is a* $(\sqrt{2m})$*-approximation to the fractional optimum.*

Proof Let $y = \{y_{i,S}\}_{(i,S)\in E}$ be the optimal fractional allocation. For every player i with $x_{i,S_i} = 1$ (for some S_i), let $Y_i = \{\ (i',S) \in E \mid y_{i',S} > 0 \text{ and } (i',S) \text{ was removed from } E \text{ when } (i,S_i) \text{ was added }\}$. We show that $\sum_{(i',S)\in Y_i} y_{i',S}w_{i',S} \leq (\sqrt{2}\sqrt{m})w_{i,S_i}$, which proves the claim. We first have:

$$\sum_{(i',S)\in Y_i} y_{i',S}w_{i',S} = \sum_{(i',S)\in Y_i} y_{i',S}\frac{w_{i',S}}{\sqrt{|S|}}\sqrt{|S|} \leq \tag{1.13}$$

$$\frac{w_{i,S_i}}{\sqrt{|S_i|}}\sum_{(i',S)\in Y_i} y_{i',S}\cdot\sqrt{|S|} \leq \frac{w_{i,S_i}}{\sqrt{|S_i|}}\sqrt{(\sum_{(i',S)\in Y_i} y_{i',S})(\sum_{(i',S)\in Y_i} y_{i',S}\cdot|S|)}$$

The first inequality follows since (i,S_i) was chosen by greedy when (i',S) was in E, and the second inequality is a simple algebraic fact. We also have:

$$\sum_{(i',S)\in Y_i} y_{i',S} \leq \sum_{j\in S_i}\ \sum_{(i',S)\in Y_i, j\in S} y_{i',S} + \sum_{(i,S)\in Y_i} y_{i,S} \leq \sum_{j\in S_i} 1 + 1 \leq |S_i| + 1 \tag{1.14}$$

where the first inequality holds since every $(i',S) \in Y_i$ has either $S \cap S_i \neq \emptyset$ or $i' = i$, and the second inequality follows from the feasibility constraints of CA-P, and,

$$\sum_{(i',S)\in Y_i} y_{i',S}\cdot|S| \leq \sum_{j\in\Omega}\ \sum_{(i',S)\in Y_i, j\in S} y_{i',S} \leq m \tag{1.15}$$

Combining 1.13, 1.14, and 1.15, we get what we need:

$$\sum_{(i',S)\in Y_i} y_{i',S}w_{i',S} \leq \frac{w_{i,S_i}}{\sqrt{|S_i|}}\cdot\sqrt{|S_i|+1}\cdot\sqrt{m} \leq \sqrt{2}\cdot\sqrt{m}\cdot w_{i,S_i}$$

□

Greedy is not truthful, but with the decomposition-based mechanism, we use randomness in order to "plug-in" truthfulness. We get:

Theorem 1.31 *The decomposition-based mechanism with Greedy as the integrality-gap verifier is individually rational and truthful-in-expectation, and obtains an approximation of* $\sqrt{2}\cdot\sqrt{m}$ *to the social welfare.*

Remarks. The decomposition-based technique is quite general, and can be used in other cases, if an integrality-gap verifier exists for the LP formulation of the problem. Perhaps the most notable case is multi-unit CAs, where there exist B copies of each item, and any player desires at most one copy from each item. In this case, one can verify a $O(m^{\frac{1}{B+1}})$ integrality gap, and this is the best-possible in polynomial time. To date, the decomposition-based mechanism is the only truthful mechanism with this tight guarantee.

Nevertheless, this method is not completely general, as VCG is. One drawback is for special cases of CAs, where low approximation ratios exist, but the integrality gap of the LP remains the same. For example, with submodular valuations, the integrality gap of CA-P is the same (the constraints do not change), but lower-than-2 approximations exist. To date, no truthful mechanism with constant approximation guarantees is known for this case. One could in principle construct a different LP formulation for this case, with a smaller integrality gap, but these attempts were unsuccessful so far.

While truthfulness-in-expectation is a natural modification of (deterministic) truthfulness, and although this notion indeed continues to be a worst-case notion, still it is inferior to truthfulness. Players are assumed to only care about their *expected* utility, and not about the variance, for example. A stronger notion is that of "universal truthfulness", were players maximize their utility for every coin toss. But even this is still weaker. While in classic algorithmic settings one can use the law of large numbers to approach the expected performance, in mechanism design one cannot repeat the execution and choose the best outcome as this affects the strategic properties. Deterministic mechanisms are still a better choice.

1.4.2 Algorithmic implementation in undominated strategies

As mentioned, randomization has problematic aspects for mechanism design, and the search for deterministic mechanisms is an important challenge. In addition, most computationally efficient mechanisms are direct revelation, but in practice, indirect mechanisms (players compete by raising prices and winners pay their last bid) are much preferred. We next discuss an interesting attempt to handle these criticisms.

Single-Value players. The mechanisms of this section fit the special case of players that desire several different bundles, all for the same value: Player i is *single-valued* if there exists $\bar{v}_i \geq 1$ such that for any bundle s, $v_i(s) \in \{0, \bar{v}_i\}$. I.e. i desires any one bundle out of a *collection* $\bar{S}_i$ of bundles, for a value $\bar{v}_i$. We denote such a player by $(\bar{v}_i, \bar{S}_i)$. $\bar{v}_i$ and $\bar{S}_i$) are private information of the player. Since $\bar{S}_i$ may be of size exponential in m, we assume the query access model, as detailed below.

An iterative wrapper. We start with a wrapper to a given algorithmic sub-procedure, which will eventually convert algorithms to a mechanisms, with a small approximation loss. It operates in iterations, with iteration index j, and maintains the tentative winners W_j, the sure-losers L_j, and a "winning bundle" s_i^j for every i. In each iteration, the sub-procedure is invoked to update the set of winners to W_{j+1} and the winning bundles to s^{j+1}. Every active non-winner then chooses to double his bid (v_i^j) or to permanently retire. This is iterated until all non-winners retire:

Definition 1.32 (The Wrapper) Initialize $j = 0$, $W_j = L_j = \emptyset$, and for every player i, $v_i^0 = 1$ and $s_i^0 = \Omega$. While $W_j \cup L_j \neq$ "all players", perform:

1. $(W_{j+1}, s^{j+1}) \leftarrow \mathrm{PROC}(v^j, s^j, W_j)$.

2. $\forall i \notin W_{j+1} \cup L_j$, i chooses whether to double his value ($v_i^{j+1} \leftarrow 2 \cdot v_i^j$) or to permanently retire ($v_i^{j+1} \leftarrow 0$). For all others set $v_i^{j+1} \leftarrow v_i^j$.

3. Update $L_{j+1} = \{i \in N \mid v_i^{j+1} = 0\}$ and $j \to j+1$, and reiterate.

Outcome: Let $J = j$ (total number of iterations). Every $i \in W_J$ gets s_i^J and pays v_i^J. All others lose (get nothing, pay 0).

For feasibility, PROC must maintain: $\forall i, i' \in W_{j+1}$, $s_i^{j+1} \cap s_{i'}^{j+1} = \emptyset$.

We need to analyze the strategic choices of the players, and the approximation loss (relative to PROC). This will be done gradually. We first worry about minimizing the number of iterations:

Definition 1.33 (Proper procedure) PROC is proper if (1) **Pareto:** $\forall i \notin W_{j+1} \cup L_j$, $s_i^{j+1} \cap (\cup_{l \in W_{j+1}} s_l^{j+1}) \neq \emptyset$, and (2) **Shrinking-sets:** $\forall i, s_i^{j+1} \subseteq s_i^j$.

A "reasonable" player will not increase v_i^j above $\bar{v}_i$, otherwise his utility will be non-positive (this strategic issue is formally discussed below). Assuming this, there will clearly be at most $n \cdot \log(v_{max})$ iterations, where $v_{max} = \max_i \bar{v}_i$. With a proper procedure this bound becomes independent of n:

Lemma 1.34 *If every player i never increases v_i^j above $\bar{v}_i$, then any proper procedure performs at most $2 \cdot \log(v_{max}) + 1$ iterations.*

Proof Consider iteration $j = 2 \cdot \log(v_{max}) + 1$, and some $i_1 \notin W_{j+1} \cup L_j$ that (by contradiction) doubles his value. By Pareto, there exists $i_2 \in W_{j+1}$ such that $s_{i_1}^{j+1} \cap s_{i_2}^{j+1} \neq \emptyset$. By "shrinking-sets", in every $j' < j$ their winning bundles intersect, hence at least one of them was not a winner, and doubled his value. But then $v_{i_1}^j \geq v_{max}$, a contradiction. □

This affects the approximation guarantee, as shown below, and also implies that the Wrapper adds only a polynomial-time overhead to PROC.

A warm-up. Consider the case of known single minded players (KSM), where a player desires *one* specific bundle, $\bar{s}_i$, which is public information (she can only lie about her value). This allows for a simple analysis: the wrapper converts any given c-approx. to a dominant-strategy mechanism with $O(\log(v_{max}) \cdot c)$ approximation. Thus, we get a *deterministic* technique to convert algorithms to mechanisms, with a small approximation loss.

Here, we initialize $s_i^0 = \bar{s}_i$, and set $s_i^{j+1} = s_i^j$, which trivially satisfies the shrinking-sets property. In addition, pareto is satisfied w.l.o.g. since if not, add winning players in an arbitrary order until pareto holds. For KSM players this takes $O(n \cdot m)$ time. Third, we need one more property:

Definition 1.35 (Improvement) $\sum_{i \in W_{j+1}} v_i^j \geq \sum_{i \in W_j} v_i^j$.

This is again without loss of generality: if the winners outputted by PROC violate this, simply output W_j as the new winners. To summarize, we use:

Definition 1.36 (The KSM-PROC) Given a c-approx. A for KSM players, KSM-PROC invokes A with s^j (the desired bundles) and v^j (player values). Then, it post processes the output to verify pareto and improvement.

Proposition 1.37 *Under dominant strategies, i retires iff $\bar{v}_i/2 \leq v_i^j \leq \bar{v}_i$.*

(the simple proof is omitted). For the approximation, the following analysis carries through to the single-value case. Let $S_i|_{s_i^j} = \{s \in S_i \mid s \subseteq s_i^j\}$, and

$$R_j(\vec{v}, \vec{S}) = \{\ (v_i, S_i|_{s_i^j})\ |\ \ i \text{ retired at iteration } j\ \}, \tag{1.16}$$

i.e. for every player i that retired at iteration j the set $R_j(\vec{v}, \vec{S})$ contains a single-value player, with value v_i (given as a parameter), and desired bundles $S_i|_{s_i^j}$ (where S_i is given as a parameter). For the KSM case, $R_j(\bar{v}, \bar{S})$ is exactly all retired players in iteration j, as the operator "$|_{s_i^j}$" has no effect. Hence, to prove the approximation, we need to bound the value of the optimal allocation to the players in $\bar{R} = \cup_{j=1}^{J} R_j(\bar{v}, \bar{S})$. For an instance X of single-value players, let $OPT(X)$ be the value of the optimal allocation to the players in X. In particular: $OPT(R_j(\vec{v}, \vec{S})) = \max_{\text{all allocations } (s_1,\ldots,s_n) \text{ s.t.} s_i \in S_i|_{s_i^j}} \{\sum_{i:\ s_i \neq \emptyset} v_i\ \}$.

Definition 1.38 (Local approximation) A proper procedure is a c-local-approximation w.r.t a strategy set D if it satisfies improvement, and, for any combination of strategies in D and any iteration j,

Algorithmic approximation $OPT(R_j(v^j, \bar{S})) \leq c \cdot \sum_{i \in W_j} v_i^j$

Value bounds $v_i^j \leq v_i(s_i^j)$, and, if i retires at j then $v_i^j \geq \bar{v}_i/2$.

Claim 1.39 *Given a c-approximation for single minded players, KSM-PROC is a c-local-approximation for the set D of dominant strategies.*

(This follows from all the above). We next translate local approximation to global approximation (this is valid also for the single-value case):

Claim 1.40 *A c-local-approximation satisfies $OPT(\bar{R}) \leq 5 \cdot \log(v_{max}) \cdot c \cdot \sum_{i \in W_J} \bar{v}_i$ whenever players play strategies in D.*

Proof By the value bounds, $OPT(R_j(\bar{v}, \bar{S})) \leq 2 \cdot OPT(R_j(v^j, \bar{S}))$. Since $OPT(R_j(v^j, \bar{S})) \leq c \cdot \sum_{i \in W_j} v_i^j$ by algorithmic approximation, $\sum_{i \in W_j} v_i^j \leq \sum_{i \in W_{j+1}} v_i^{j+1}$ by improvement, and $v_i^J \leq \bar{v}_i$ (again by the value bounds), we get $OPT(R_j(\bar{v}, \bar{S})) \leq 2 \cdot c \cdot \sum_{i \in W_J} \bar{v}_i$. Hence $OPT(\bar{R}) \leq \sum_{j=1}^{J} OPT(R_j(\bar{v}, \bar{S})) \leq J \cdot 2 \cdot c \cdot \sum_{i \in W_J} \bar{v}_i$. Since $J \leq 2 \cdot \log(v_{max}) + 1$, the claim follows. □

For single-minded players, $\bar{R}$ is the set of losing players, hence we conclude:

Theorem 1.41 *Given any c-approx. for KSM players, the Wrapper with KSM-PROC implements an $O(log(v_{max}) \cdot c)$ approx. in dominant strategies.*

A sub-procedure for single-value players. Two assumptions are relaxed: players are now multi-minded, and their desired bundles are unknown. Here, we define the following specific sub-procedure. For a set of players X, let Free(X, s^{j+1}) denote the items not in $\cup_{i \in X} s_i^j$.

Definition 1.42 (1-CA-PROC) Let $M_j = \text{argmax}_{i \in N}\{v_i^j\}$, $GREEDY_j = \emptyset$. For every player i with $v_i^j > 0$, in descending order of values, perform:

- **Shrinking the winning set:** If $i \notin W_j$ allow him to pick a bundle $s_i^{j+1} \subseteq$ Free$(GREEDY_j, s^{j+1}) \cap s_i^j$ such that $|s_i^{j+1}| \leq \sqrt{m}$. In any other case ($i \in W_j$ or i does not pick) set $s_i^{j+1} = s_i^j$.
- **Updating the current winners:** If $|s_i^{j+1}| \leq \sqrt{m}$, add i to any of the allocations $W \in \{W_j, M_j, GREEDY_j\}$ for which $s_i^{j+1} \subseteq$ Free(W, s^{j+1}).

Output s^{j+1} and $W \in \{W_j, M_j, GREEDY_j\}$ that maximizes $\sum_{i \in W} v_i^j$.

Recall that the non-winners then either double their value or retire, and we reiterate. This is the main conceptual difference from "regular" direct revelation mechanisms: here, the players themselves gradually determine their winning set (focusing on one of their desired bundles), and their price. Intuitively, it is not clear how should a "reasonable" player shrink his winning set, when approached. Ideally, a player should focus on a desired bundle that intersects few, low-value competitors. But in early iterations this information is not available. Thus there is no clear-cut on how to shrink the winning set, and the resulting mechanism does not contain a dominant strategy. This brings up the need for a new solution concept.

Algorithmic implementation in undominated strategies. We would like to allow the mechanism designer to "leave in" several strategic choices, but to require the approximation *in each of these choices.*

Definition 1.43 (Algorithmic implementation) A mechanism M is an algorithmic implementation of a c-approximation in undominated strategies if there exists a set of strategies, D, such that (i) M obtains a c-approximation for any combination of strategies from D, in polynomial time, and (ii) For any strategy not in D, there exists a strategy in D that weakly dominates it, and this transition is polynomial-time computable.

The important ingredients of dominant-strategies implementation are here: the only assumption is that a player is willing to replace any chosen strategy with a strategy that dominates it. Indeed, this guarantees at least the same utility, even in the worst-case, and by definition can be done in polynomial time. In addition, again as in dominant-strategy implementability, this notion does not require any form of coordination among the players (unlike Nash equilibrium), or that players have any assumptions on the rationality of the others (as in "iterative deletion of dominated strategies"). However, two differences from dominant-strategies implementation are worth mentioning: (I) A player might regret his chosen strategy, realizing in retrospect that another strategy from D would have performed better, and (II) Deciding how to play is not straight-forward. While a player will not end up playing a strategy that does not belong to D, it is not clear how will he choose one of the strategies of D. This may depend for example on the player's own beliefs about the other players, or on the computational power of the player.

Another remark, about the connection to implementation in undominated strategies, is in place. The definition of D *does not* imply that all undominated strategies belong to D, but rather that for every undominated strategy, there is an *equivalent* strategy inside D (i.e. a strategy that yields the same utility, no matter what the others play). The same problem occurs with dominant-strategy implementations, e.g. VCG, where it is not required that truthfulness should be the *only* dominant strategy, just *a* dominant strategy.

Analysis. We proceed by characterizing the required set D of strategies. We say that player i is "loser-if-silent" at iteration j if, when asked to shrink her bundle by 1-CA-PROC, $v_i^j \geq \bar{v}_i/2$ (*retires if losing*), $i \notin W_j$ and $i \notin M_j$ (*not a winner*), and $s_i^j \cap (\cup_{i' \in W_j} s_{i'}^{j+1}) \neq \emptyset$ and $s_i^j \cap (\cup_{i' \in M_j} s_{i'}^{j+1}) \neq \emptyset$ (*remains a loser after pareto*). In other words, a loser-if-silent loses (regardless of the others' actions) unless she shrinks her winning set. Let D be all strategies that satisfy, in every iteration j:

(i) $v_i^j \leq v_i(s_i^j)$, and, if i retires at j then $v_i^j \geq \bar{v}_i/2$.
(ii) If i is "loser-if-silent" then she declares a valid desired bundle s_i^{j+1}, if such a bundle exists.

There clearly exists a (poly-time) algorithm to find a strategy $st' \in D$ that dominates a given strategy st. Hence D satisfies the second requirement of algorithmic implementation. It remains to show that the approximation is achieved for *every* combination of strategies from D.

Lemma 1.44 *1-CA-PROC is an $O(\sqrt{m})$-local-approximation w.r.t. D.*

Proof (sketch). The pareto, improvement, and value-bounds properties are immediate from the definition of the procedure and the set D. The $O(\sqrt{m})$-algorithmic-approximation property follows from the following argument. We need to bound $OPT = OPT(\{(v_i^j, \bar{S}_i|_{s_i^j}) \mid i \text{ retired at iteration } j\})$ by the sum of values of the players in W_{j+1}. We divide the winners in OPT to four sets. Those that are in M_j, $GREEDY_j$, W_j, or in non of the above. For the first three sets the 1-CA-PROC explicitly verifies our need. It remains to handle players in the forth set. First notice that such a player is loser-if-silent. If such a player receives in OPT a bundle with size at least $\sqrt{m}$ we match him to the player with the highest value in M_j. There can be at most $\sqrt{m}$ players in OPT with bundles of size at least $\sqrt{m}$, so we lose a $\sqrt{m}$ factor for these players. If a player, i, in the forth set, receives in OPT a bundle with size at most $\sqrt{m}$, let s_i^* be that bundle. Since he is a loser-if-silent, there exists $i' \in GREEDY_j$ such that $s_{i'}^j \cap s_i^* \neq \emptyset$ and $v_i^j \leq v_{i'}^j$. We map i to i'. For any i_1, i_2 that were mapped to i' we have that $s_{i_1}^* \cap s_{i_2}^* = \emptyset$ since both belong to OPT. Since the size of $s_{i'}^j$ is at most $\sqrt{m}$ it follows that at most $\sqrt{m}$ players can be mapped to i', so we lose a $\sqrt{m}$ factor for these players as well. This completes the argument. □

In the single-value case, $\bar{R}$ does not contain all players, so we cannot repeat the argument from the KSM case that immediately linked local approximation and global approximation. However, Claim 1.40 still holds, and we use $\bar{R}$ as an intermediate set of "virtual" players. The link to the true players is as follows (recall that m denotes the number of items):

Definition 1.45 (First time shrink) PROC satisfies "first time shrink" if for any $i_1, i_2 \in \{i \ : \ |s_i^j| = m \ \& \ |s_i^{j+1}| < m\}$, $s_{i_1}^{j+1} \cap s_{i_2}^{j+1} = \emptyset$.

1-CA-PROC satisfies this since any player that shrinks his winning bundle is added to $GREEDY_j$.

Lemma 1.46 *Given a c-local-approximation (w.r.t. D) that satisfies first-time-shrink, the Wrapper obtains an $O(\log^2(v_{max}) \cdot c)$ approximation for any profile of strategies in D.*

Proof We continue to use the notation of claim 1.40. Let $P = \{(\bar{v}_i, \bar{S}_i) \ :$ i lost, and $|s_i^J| < m\}$. Players in P appear with all their desired bundles, while players in $\bar{R}$ appear with only part of their desired bundles. However, ignoring the extra bundles in P incurs only a bounded loss:

Claim 1.47 $OPT(P) \leq J \cdot OPT(\bar{R})$.

Proof Define P_j to be all players in P that first shrank their bundle at iteration j. By "first time shrink", and since winning bundles only shrink, $s^j_{i_1} \cap s^j_{i_2} = \emptyset$ for every $i_1, i_2 \in P_j$. Therefore $OPT(\bar{R}) \geq \sum_{i \in P_j} \bar{v}_i$: every player i in P_j corresponds to a player in $\bar{R}$, and all these players have disjoint bundles in $\bar{R}$ since the bundles of i are contained in s^j_i. We also trivially have $OPT(P_j) \leq \sum_{i \in P_j} \bar{v}_i$. Thus, for any j, $OPT(P_j) \leq OPT(\bar{R})$, and $OPT(P) \leq \sum_j OPT(P_j) \leq J \cdot OPT(\bar{R})$. □

To prove the lemma, first notice that all true players are contained in $P \cup \bar{R} \cup W_J$: all retiring players belong to $\bar{R} \cup P$ (if a player shrank his bundle then he belongs to P with all his true bundles, and if a player did not shrink his bundle at all then he belongs to $\bar{R}$ with all his true bundles) and all non-retiring players belong to W_J. From the above we have $OPT(P \cup \bar{R}) \leq OPT(P) + OPT(\bar{R}) \leq J \cdot OPT(\bar{R}) + OPT(\bar{R}) \leq 4 \cdot J^2 \cdot c \cdot \sum_{i \in W_J} \bar{v}^J_i$. Since s^J_i contain some desired bundle of player i, we have that $OPT(W_J) = \sum_{i \in W_J} \bar{v}_i$. Thus we get that $OPT(P \cup \bar{R} \cup W_J) \leq 5 \cdot J^2 \cdot \tilde{c} \cdot \sum_{i \in W_J} \bar{v}^J_i$. Since $J \leq 2 \cdot \log(v_{max}) + 1$ by Lemma 1.34, the lemma follows. □

By all the above, we conclude:

Theorem 1.48 *The Wrapper with 1-CA-PROC is an algorithmic implementation of an $O(\log^2(v_{max}) \cdot c)$-approximation for single-value players.*

1.4.3 A broader view

The search for truthful CAs is an active field of research. Roughly speaking, two techniques have been proved useful for constructing truthful CAs. In "Maximal-In-Range" mechanisms, the range of possible allocations is restricted, and the optimal-in-this-range allocation is chosen. This achieves deterministic truthfulness with an $O\sqrt{m})$-approx. for subadditive valuations [DNS05], an $O(\frac{m}{\sqrt{\log m}})$-approx. for general valuations [HKDMT04], and a 2-approx. when all items are identical ("multi-unit auctions") [DN06]. A second technique is to partition the set of players, sample statistics from one set, and use it to obtain a good approximation for the other. See Chapter 13 for details. This technique obtains an $O(\sqrt{m})$-approx. for general valuations, and an $O(\log^2 m)$ for XOS valuations [DNS06]. The truthfulness here is "universal", i.e. for any coin toss – a stronger notion than truthfulness in expectation. [BGN03] use a similar idea to obtain a truthful and *deterministic* $O(B \cdot m^{\frac{1}{B-2}})$-approx. for multi-unit CAs with $B \geq 3$ copies of each item. For special cases of CAs, these techniques do not yet manage to obtain constant-factor truthful approximations ([DN06] prove this impossibility for Maximal-In-Range mechanisms). Due to the importance of constant-factor approximations, explaining this gap is challenging:

Open Question Does there exist truthful constant-factor approximations for special cases of CAs?

For general valuations, the above shows a significant gap in the power of randomized vs. deterministic techniques. It is not known if this gap is essential. A possible argument for this gap is that, for general valuations, every deterministic mechanism is VCG-based, and these have no power. [LMN03] have initiated an investigation for the first part of the argument, obtaining only partial results. [DN06] have studied the other part of the argument, again with only partial results.

Open Question What are the limitations of deterministic truthful CAs? Does approximation and dominant-strategies clash in some fundamental and well-defined way for CAs?

In light of this, examining different solution concepts for deterministic mechanisms seems natural. We described one such attempt. [KPS05] describe an "almost truthful" deterministic FPAS for multi-unit auctions. [LN05] define a notion of "Set-Nash" for multi-unit auctions in an online setting, and prove such a gap: for their setting, deterministic truthfulness obtains significantly lower approximations than Set-Nash implementations. However Set-Nash is a rather weak notion, and an interesting question is whether the notion of algorithmic implementation (discussed above) can similarly help:

Open Question Does there exist a domain in which a computationally-efficient algorithmic implementation achieves a better approximation than any computationally-efficient dominant-strategy implementation?

This section was devoted to welfare maximization. Revenue maximization is another important goal for CA design. The mechanism of [BGN03] obtains the same guarantees with respect to the optimal revenue. More tight results for multi-unit auctions with budget constrained players are given by [BCI$^+$05], and for unlimited-supply CAs by [BBHM05]. These are preliminary results for special cases; this issue is still quite unexplored.

1.5 Impossibilities of Dominant Strategy Implementability

We saw an interesting contrast between deterministic and randomized truthfulness, and asked whether the source of this difficulty can be rigorously identified and characterized. This question has remained mostly unexplored, with few exceptions, one of which we review next. Recall the social choice setting of section 1.2.1, let $V = V_1 \times \cdots \times V_n$, and assume that $f : V \to A$ is onto A. VCG implements welfare-maximization (see Chapter 9), for any domain. This extends to *weighted* welfare maximizers:

Definition 1.49 (Affine maximizer) f is an "affine maximizer" if there exist weights $k_1, \dots, k_n$ and $\{C_x\}_{x \in A}$ such that, for all $v \in V$,

$$f(v) \in \text{argmax}_{x \in A} \ \{ \ \Sigma_{i=1}^{n} k_i v_i(x) + C_x \ \}.$$

Other function forms may be desirable, in order to e.g. maximize the revenue, employ different fairness criteria, or to consider the computational complexity of the function. The fundamental question is what other function forms are implementable. If the domain is unrestricted, the answer is sharp:

Theorem 1.50 *Suppose $|A| \geq 3$ and $V_i = \Re^A$ for all i. Then f is dominant-strategy implementable iff it is an affine maximizer.*

A generalization of VCG arguments shows that any affine maximizer is implementable. In this section we prove an easier version of the other direction. The proof is simplified by adding an extra requirement, but the essential structure is kept. The exercises give guidelines to complete the full proof.

Definition 1.51 (Neutrality) f is neutral if for all $v \in V$, if there exists an alternative x such that $v_i(x) > v_i(y)$, for all i and $y \neq x$, then $f(v) = x$.

Neutrality essentially implies that if a function is indeed an affine maximizer then the additive constants C_x are all zero.

Theorem 1.52 *Suppose $|A| \geq 3$ and for every i, $V_i = \Re^A$. If f is dominant-strategy implementable and neutral then it must be an affine maximizer.*

For the proof, we start with two monotonicity conditions:

Definition 1.53 (Positive Association of Differences (PAD)) f satisfies PAD if the following holds for any $v, v' \in V$. Suppose $f(v) = x$, and for any $y \neq x$, and any i, $v'_i(x) - v_i(x) > v'_i(y) - v_i(y)$. Then $f(v') = x$.

Claim 1.54 *Any implementable function f, on any domain, satisfies PAD.*

Proof Let $v^i = (v'_1, \dots, v'_i, v_{i+1}, \dots, v_n)$, i.e. players up to i declare according to v'; the rest declare according to v. Thus $v^0 = v$, $v^n = v'$, and $f(v^0) = x$. Suppose $f(v^{i-1}) = x$ for some $1 \leq i \leq n$. For every alternative $y \neq x$ we have $v^i_i(y) - v^{i-1}_i(y) < v^i_i(x) - v^{i-1}_i(x)$, and in addition $v^{i-1}_{-i} = v^i_{-i}$. Thus, W-MON implies that $f(v^i) = x$. By induction, $f(v^n) = x$. □

In an unrestricted domain, W-MON can be generalized as follows:

Definition 1.55 (Generalized-WMON) For every $v, v' \in V$ with $f(v) = x$ and $f(v') = y$ there exists a player i such that: $v'_i(y) - v_i(y) \geq v'_i(x) - v_i(x)$.

In W-MON, we fix a player and fix the declarations of the others. Here this qualifier is dropped. Another way of looking at this property is the following: If $f(v) = x$ and $v'(x) - v(x) > v'(y) - v(y)$ then $f(v') \neq y$ (a word about notation: for $\alpha, \beta \in \Re^n$, we use $\alpha > \beta$ to denote that $\forall i,\ \alpha_i > \beta_i$).

Claim 1.56 *If the domain is unrestricted and f is implementable then f satisfies Generalized-WMON.*

Proof Fix any v, v'. We show that if $f(v') = x$ and $v'(y) - v(y) > v'(x) - v(x)$ for some $y \in A$ then $f(v) \neq y$. By contradiction, suppose $f(v) = y$. Fix $\Delta \in \Re^n$ such that $v'(x) - v'(y) = v(x) - v(y) - \Delta$, and define v'':

$$\forall i,\ z \in A\ :\ v''_i(z) = \begin{cases} \min\{v_i(z)\ ,\ v'_i(z) + v_i(x) - v'_i(x)\} - \Delta_i & z \neq x, y \\ v_i(x) - \frac{\Delta_i}{2} & z = x \\ v_i(y) & z = y. \end{cases}$$

By PAD, the transition $v \to v''$ implies $f(v'') = y$, and the transition $v' \to v''$ implies $f(v'') = x$, a contradiction. $\square$

We now get to the main construction. For any $x, y \in A$, define:

$$P(x,y) = \{\alpha \in \Re^n \mid \exists v \in V\ :\ v(x) - v(y) = \alpha,\ f(v) = x\ \}. \tag{1.17}$$

Looking at differences helps since we need to show that $\sum_i k_i[v_i(x) - v_i(y)] \geq C_y - C_x$ if $f(v) = x$. Note that $P(x,y)$ is not empty (by assumption there exists $v \in V$ with $f(v) = x$), and that if $\alpha \in P(x,y)$ then for any $\delta \in \Re^n_{++}$ (i.e. $\delta > \vec{0}$), $\alpha + \delta \in P(x,y)$: take v with $f(v) = x$ and $v(x) - v(y) = \alpha$, and construct v' by increasing $v(x)$ by δ, and setting the other coordinates as in v. By PAD $f(v') = x$, and $v'(x) - v'(y) = \alpha + \delta$.

Claim 1.57 *For any $\alpha, \epsilon \in \Re^n$, $\epsilon > \vec{0}$: (i) $\alpha - \epsilon \in P(x,y) \Rightarrow -\alpha \notin P(y,x)$, and (ii) $\alpha \notin P(x,y) \Rightarrow -\alpha \in P(y,x)$.*

Proof (i) Suppose by contradiction that $-\alpha \in P(y,x)$. Therefore there exists $v \in V$ with $v(y) - v(x) = -\alpha$ and $f(v) = y$. As $\alpha - \epsilon \in P(x,y)$, there also exists $v' \in V$ with $v'(x) - v'(y) = \alpha - \epsilon$ and $f(v') = x$. But since $v(x) - v(y) = \alpha > v'(x) - v'(y)$, this contradicts Generalized-WMON. (ii) For any $z \neq x, y$ take some $\beta_z \in P(x,z)$ and fix some $\epsilon > \vec{0}$. Fix some v such that $v(x) - v(y) = \alpha$ and $v(x) - v(z) = \beta_z + \epsilon$ for all $z \neq x, y$. By the above argument, $f(v) \in \{x, y\}$. Since $v(x) - v(y) = \alpha \notin P(x,y)$ it follows that $f(v) = y$. Thus $-\alpha = v(y) - v(x) \in P(y,x)$, as needed. $\square$

Claim 1.58 *Fix $\alpha, \beta, \epsilon_1, \epsilon_2, \in \Re^n$, $\epsilon_i > \vec{0}$, such that $\alpha - \epsilon_1 \in P(x,y)$ and $\beta - \epsilon_2 \in P(y,z)$. Then $\alpha + \beta - (\epsilon_1 + \epsilon_2)/2 \in P(x,z)$.*

Proof For any $w \neq x, y, z$ fix some $\delta_w \in P(x,w)$. Choose any v such that $v(x) - v(y) = \alpha - \epsilon_1/2$, $v(y) - v(z) = \beta - \epsilon_2/2$, and $v(x) - v(w) = \delta_w + \epsilon$ for all $w \neq x, y, z$ (for some $\epsilon > \vec{0}$). By Generalized-WMON, $f(v) = x$. Thus $\alpha + \beta - (\epsilon_1 + \epsilon_2)/2 = v(x) - v(z) \in P(x,z)$. □

Claim 1.59 *If α is in the interior of $P(x,y)$ then α is in the interior of $P(x,z)$, for any $z \neq x, y$.*

Proof Suppose $\alpha - \epsilon \in P(x,y)$ for some $\epsilon > \vec{0}$. By neutrality we have that $\epsilon/4 - \epsilon/8 = \epsilon/8 \in P(y,z)$. By claim 1.58 we now get that $\alpha - \epsilon/4 \in P(x,z)$, which implies that α is in the interior of $P(x,z)$. □

By similar arguments, we also have that if α is in the interior of $P(x,z)$ then α is in the interior of $P(w,z)$. Thus we get that for any $x, y, w, z \in A$, not necessarily distinct, the interior of $P(x,y)$ is equal to the interior of $P(w,z)$. Denote the interior of $P(x,y)$ as P.

Claim 1.60 *P is convex.*

Proof We show that $\alpha, \beta \in P$ implies $(\alpha + \beta)/2 \in P$. A known fact from convexity theory then implies that P is convex †. By claim 1.58, $\alpha + \beta \in P$. We show that for any $\alpha \in P$ we have $\alpha/2 \in P$ as well, which then implies the claim. Suppose by contradiction that $\alpha/2 \notin P$. Thus by claim 1.57, $-\alpha/2 \in P$. Then $\alpha/2 = \alpha + (-\alpha/2) \in P$, a contradiction. □

We now conclude the proof of theorem 1.52. Neutrality implies that $\vec{0}$ is on the boundary of any $P(x,y)$, hence it is not in P. Let $\bar{P}$ denote the closure of P. By the separation lemma, there exists a $k \in \Re^n$ such that for any $\alpha \in \bar{P}$, $k \cdot \alpha \geq 0$. Suppose $f(v) = x$ for some $v \in V$, and fix any $y \neq x$. Thus $v(x) - v(y) \in P(x,y)$, and $k \cdot v(x) - v(y) \geq 0$. Hence $k \cdot v(x) \geq k \cdot v(y)$, and the theorem follows.

1.6 Bibliographic Notes

Cyclic monotonicity was formalized by Rochet [Roc87]. The graph representation method is due to Gui, Muller, and Vohra [GMV04]. Weak monotonicity was independently defined by Bikhchandani, Chatterjee, and Sen [BCS03], and by Lavi, Mu'alem, and Nisan [LMN03], who also gave the proof for order-based domains. Some of these results were published

† For $\alpha, \beta \in P$ and $0 \leq \lambda \leq 1$, build a series of points that approach $\lambda\alpha + (1-\lambda)\beta$, such that any point in the series has a ball of some fixed radius around it that fully belongs to P.

in [BCL$^+$06]. Saks and Yu [SY05] give a generalization to convex domains. The connection between classic scheduling and mechanism design was suggested by Nisan and Ronen [NR01], where they studied unrelated machines and reached mainly impossibilities. Archer and Tardos studied the case of related machines, and Section 1.3 is based on their work [AT01]. Deterministic mechanisms for the problem have been suggested since then by several others, and the current best approximation ratio, 3, is given by Kovacs [Kov05]. Section 1.4.1 is based on the work of Lavi and Swamy [LS05], and section 1.4.2 is based on the work of Babaioff, Lavi, and Pavlov [BLP06]. Roebrts [Rob79] characterized dominant strategy implementability for unrestricted domains. The proof given here is based on [LMN04]. Generalized-WMON was suggested in [LMN03], which explored the same characterization question for restricted domains.

Exercises

1.1 (Weak Monotonicity in a simple convex domain) (a) Suppose that $A = \{a_1, a_2, ..., a_k\}$, and a given f satisfies W-MON, plus, for any $1 \leq i \leq k-1$ and any $x \in A$, the following conditions: (i) $\delta_{a_i,a_{i+1}} + \delta_{a_{i+1},a_i} = 0$, (ii) $\delta_{a_i,x} + \delta_{x,a_{i+1}} \geq \delta_{a_i,a_{i+1}}$, and (iii) $\delta_{a_{i+1},x} + \delta_{x,a_i} \geq \delta_{a_{i+1},a_i}$. Show that f satisfies cyclic monotonicity. (b) Fix two valuations $v_i, v'_i \in \Re^{|A|}$, and construct a domain V_i which is the convex hull of v_i, v'_i. I.e. $v''_i \in V_i$ if and only if there exists $0 \leq \lambda \leq 1$ such that $v''_i(x) = \lambda v_i(x) + (1-\lambda) v'_i(x)$ for every $x \in A$. Show that if $f : V \to A$ satisfies W-MON, then there exists a complete order over A such that the conditions of (a) are all satisfied.

1.2 (Scheduling unrelated machines) In the model of unrelated machines, each job j creates a load p_{ij} on each machine i, where the loads are completely unrelated. Prove, using W-MON, that no truthful mechanism can approximate the makespan with a factor better than two. Hint: start with four jobs that have $p_{ij} = 1$ for all i, j.

1.3 A deterministic greedy rounding of the fractional scheduling 1.16 assigns each job in full to the first machine that got a fraction of it. Explain why this is a 2-approximation, and show by an example that this violates monotonicity.

1.4 Prove that 1-CA-PROC of Def. 1.42, and Greedy for multi-minded players of Def. 1.29 are not dominant-strategy implementable.

1.5 (Converting algorithms to mechanisms) Fix an alternative set A, and suppose that for any player i, there is a fixed, known subset $A_i \subset A$, such that a valid valuation assigns some positive real number in

$[v_{min}, v_{max}]$ to every alternative in A_i, and zero to the other alternatives. Suppose v_{min} and v_{max} are known. Given a c-approximation algorithm to the social welfare for this domain, construct a randomized truthful mechanism that obtains a $O(\log(v_{max}/v_{min}) \cdot c)$ approximation to the social welfare. (Hint: choose a threshold price, uniformly at random). Is this construction still valid when the sets A_i are unknown? (If not, show a counter example).

1.6 Describe a domain for which there exists an implementable social choice function that does not satisfy Generalized-WMON.

1.7 Describe a deterministic CA for general valuations that is not an affine maximizer.

1.8 This exercise aims to complete the characterization of section 1.5: Let $\gamma(x,y) = inf\{p \in \Re \mid p \cdot \vec{1} \in P(x,y) \}$. Show that $\gamma(x,y)$ is well-defined, that $\gamma(x,y) = -\gamma(y,x)$, and that $\gamma(x,z) = \gamma(x,y) + \gamma(y,z)$. Let $C(x,y) = \{\alpha - \gamma(x,y) \cdot \vec{1} \mid \alpha \in P(x,y) \}$. Show that for any $x, y, w, z \in A$, the interior of $C(x,y)$ is equal to the interior of $C(w,z)$. Use this to show that $C(x,y)$ is convex.

Conclude, by the separation lemma, that f is an affine maximizer (give an explicit formula for the additive terms C_x).

Bibliography

[AT01] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2001.

[BBHM05] M. Balcan, A. Blum, J. Hartline, and Y. Mansour. Mechanism design via machine learning. In *Proc. of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*, 2005.

[BCI$^+$05] C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. In *Proc. of the 6th ACM Conference on Electronic Commerce (ACM-EC)*, 2005.

[BCL$^+$06] S. Bikhchandani, S. Chatterjee, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 2006. To appear.

[BCS03] S. Bikhchandani, S. Chatterjee, and A. Sen. Incentive compatibility in multi-unit auctions, 2003. Working paper.

[BGN03] Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi-unit combinatorial auctions. In *Proc. of the 9th Conference of Theoretical Aspects of Rationality and Knowledge (TARK)*, 2003.

[BLP06] M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and implementation in undominated strategies. In *Proc. of the 17th Symposium on Discrete Algorithms (SODA)*, 2006.

[DN06] S. Dobzinski and N. Nisan. Approximations by computationally-efficient vcg-based mechanisms, 2006. Working paper.

[DNS05] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proc. of the 37th ACM Symposium on Theory of Computing (STOC)*, 2005.

[DNS06] S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proc. of the 38th ACM Symposium on Theory of Computing (STOC)*, 2006.

[GMV04] H. Gui, R. Muller, and R. V. Vohra. Characterizing dominant strategy mechanisms with multi-dimensional types, 2004. Working paper.

[HKDMT04] R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Economic Behavior*, 47:104–123, 2004.

[Kov05] A. Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In *Proc. of the 13th Annual European Symposium on Algorithms (ESA)*, 2005.

[KPS05] A. Kothari, D. Parkes, and S. Suri. Approximately-strategyproof and tractable multi-unit auctions. *Decision Support Systems*, 39:105–121, 2005.

[LMN03] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. of the 44th Annual Symposium on Foundations of Computer Science (FOCS)*, 2003.

[LMN04] R. Lavi, A. Mu'alem, and N. Nisan. Two simplified proofs for roberts' theorem, 2004. Working paper.

[LN05] R. Lavi and N. Nisan. Online ascending auctions for gradually expiring items. In *Proc. of the 16th Symposium on Discrete Algorithms (SODA)*, 2005.

[LS05] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*, 2005.

[NR01] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[Rob79] K. Roberts. The characterization of implementable choice rules. In Jean-Jacques Laffont, editor, *Aggregation and Revelation of Preferences.*, pages 321–349. North-Holland, 1979.

[Roc87] J. C. Rochet. A necessary and sufficient condition for rationalizability in a quasilinear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[SY05] M. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proc. of the 6th ACM Conference on Electronic Commerce (ACM-EC)*, 2005.