CHAPTER 12

Computationally Efficient Approximation Mechanisms

Ron Lavi

Abstract

We study the integration of game theoretic and computational considerations. In particular, we study the design of computationally efficient *and* incentive compatible mechanisms, for several different problem domains. Issues like the dimensionality of the domain, and the goal of the algorithm designer, are examined by providing a technical discussion on four results: (i) approximation mechanisms for single-dimensional scheduling, where truthfulness reduces to a simple monotonicity condition; (ii) randomness as a tool to resolve the computational vs. incentives clash for Combinatorial Auctions, a central multidimensional domain where this clash is notable; (iii) the impossibilities of deterministic dominant-strategy implementability in multidimensional domains; and (iv) alternative solution concepts that fit worst-case analysis, and aim to resolve the above impossibilities.

12.1 Introduction

Algorithms in computer science, and *Mechanisms* in game theory, are very close in nature. 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 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 certain 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.

The possibility of constructing mechanisms with desirable computational properties turns out to be strongly related to the *dimensionality* of the problem domain. In single-dimensional domains, the requirement for game-theoretic truthfulness reduces to a convenient algorithmic monotonicity condition that leaves ample flexibility for the algorithm designer. We demonstrate this in Section 12.2, were we study the construction of computationally efficient approximation mechanisms for the classic machine scheduling problem. Although there exists a rich literature on approximation algorithms for this problem domain, quite remarkably none of these classic results satisfy the desired game-theoretic properties. We show that when the scheduling problem is single-dimensional, then this clash is not fundamental, and can be successfully resolved.

The problem domain of job scheduling has one additional interesting aspect that makes it worth studying: it demonstrates a key difference between economics and computer science, namely the *goals* of algorithms vs. the goals of classic mechanisms. 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, existing techniques for incentive-compatible mechanism design do not fit such an objective (and, on the other hand, most existing algorithmic solutions do not yield the desired incentives). The resolution of these clashes has led to insightful techniques, and the technical exploration of Section 12.2 serves as an example.

As opposed to single-dimensional domains, *multi*-dimensionality seems to pose much harder obstacles. In Chapter 9, the monotonicity conditions that characterize truthfulness for multidimensional domains were discussed, but it seems that these conditions do not translate well to algorithmic constructions. This issue will be handled in the rest of the chapter, and will be approached in three different ways: we will explore the inherent impossibilities that the required monotonicity conditions cast on deterministic algorithmic constructions, we will introduce randomness to solve these difficulties, and we will consider alternative notions to the solution concept of truthfulness.

Our main example for a multidimensional domain will be the domain of combinatorial auctions (CAs). Chapter 11 studies CAs mostly from a computational point of view, and in contrast our focus is on designing computationally efficient *and* incentive compatible CAs. This demonstrates a second key difference between economics and computer science, namely the requirement for computational efficiency. Even if our goal is the classic economic goal of welfare maximization, we cannot use Vickrey–Clarke–Groves mechanisms (which classically implement this goal) since in many cases they are computationally inefficient. The domain of CAs captures exactly this point, and the need for computationally efficient techniques that translate algorithms to mechanisms is central. In Section 12.3 we will see how randomness can help. We describe a rather general technique that uses randomness and linear programming in order to convert algorithms to truthful-in-expectation mechanisms. Thus we get a positive answer to the computational clash, by introducing randomness.

In Section 12.4 we return to deterministic settings and to the classic definition of deterministic truthfulness, and study the impossibilities associated with it. Our motivating question is whether the three requirements (i) deterministic truthfulness, (ii) computational efficiency, and (iii) nontrivial approximation guarantees, clash in a fundamental and well-defined way. We already know that single dimensionality does not exhibit such a clash, and in this section we describe the other extreme. If a domain has full dimensionality (in a certain formal sense, to be discussed in the section body), then any truthful mechanism must be VCG. It is important to remark that this result further emphasizes our lack of knowledge about the state of affairs for all the *intermediate* range of multidimensional domains, to which CAs and its different variants belong.

As was motivated in previous chapters, the game-theoretic quest should start with the solution concept of "implementation in dominant strategies," and indeed most of this chapter follows this line of thought. However, to avoid the impossibilities mentioned earlier, we have to deepen our understandings about the alternatives at hand. Studies in economics usually turn to the solution concept of Bayesian–Nash that requires strong distributional assumptions, namely that the input distributions are known, and, furthermore, that they are *commonly* known, and agreed upon. Such assumptions seem too strong for CS settings, and criticism about these assumptions have been also raised by economists (e.g., "Wilson's doctrine"). We have already seen that randomization, and truthful-in-expectation in particular, can provide a good alternative. We conclude the chapter by providing an additional example, of a deterministic alternative solution concept, and describe a deterministic CA that uses this notion to provide nontrivial approximation guarantees.

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 significantly different structure, which causes the above-mentioned 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, and the reader is referred to, e.g., Chapter 13 for a broader discussion.

12.2 Single-Dimensional Domains: 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 make sense:

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 $l_i = \sum_{j \mid j \text{ is assigned to } i} p_j$ 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 selfish entity, incurring 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 12.1 (single-dimensional linear domains) A domain V_i of player i is single-dimensional and linear if there exist nonnegative 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, as 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 nonnegative.

We aim to design a computationally efficient approximation algorithm, *that is also implementable*. As the social goal is a certain min–max criterion, and not to minimize the *sum* of costs, we cannot use the general VCG technique. Since we have a convex domain, Chapter 9 tells us that we need a "weakly monotone" algorithm. But what exactly does this mean? Luckily, the formulation of weak monotonicity can be much simplified for single-dimensional domains.

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 the weak monotonicity condition from Chapter 9 reduces to $-q_i(c')(c' - c) \geq -q_i(c)(c' - c)$, which holds iff $q_i(c') \leq q_i(c)$. Hence from Chapter 9 we know that such an algorithm is implementable if and only if its load functions are monotone nonincreasing. Figure 12.1 describes this, and will help us figure out the required prices for implementability.

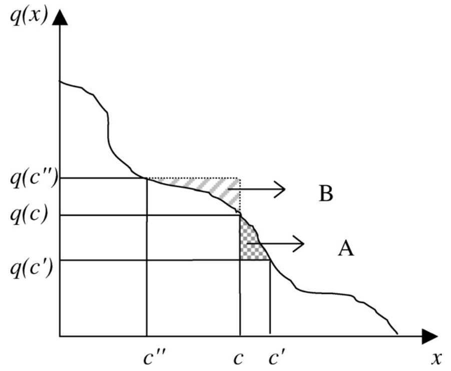

Figure 12.1. A monotone load curve.

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 Figure 12.1, we can easily verify that these prices lead to incentive compatibility: Suppose that 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 nonnegative, exactly $\int_c^\infty q_i(x)\,dx$. From all the above we get the following theorem.

Theorem 12.2 *An algorithm for a single-dimensional linear domain is implementable if and only if its load functions are nonincreasing. 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 construct a randomized mechanism, as well as a deterministic one. In the randomized case, we 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.

12.2.1 A Monotone Algorithm for the Job Scheduling Problem

Now that we understand the exact form of an implementable algorithm, we can construct one that approximates the optimal outcome. In fact, the optimum itself is implementable, since it can satisfy weak monotonicity (see the exercises for more details), but the computation of the optimal outcome is NP-hard. We wish to construct efficiently computable mechanisms, and hence design a monotone and polynomial-time approximation algorithm. Note that we face a "classic" algorithmic problem – no game-theoretic issues are left for us to handle.

Before we start, let us 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 $1, \ldots, j$ 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 $1, \ldots, j$ to machines $1, \ldots, i(j, T)$. From space considerations, it immediately follows that

$$T \geq \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i(j,T)} s_l}. \tag{12.1}$$

Now define

$$T_j = \min_i \max \left\{ \frac{p_j}{s_i}, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} s_l} \right\} \tag{12.2}$$

Lemma 12.3 *For any job-index j, the optimal makespan is at least T_j.*

PROOF Fix any $T < T_j$. We prove that T violates 12.1, 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 12.1 for each case separately.

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 12.1, 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. In addition, $i(j, T) < i_j$ 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:

$$T_{\text{LB}} = \max_j T_j \tag{12.3}$$

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

A fractional 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 12.4 (The fractional allocation) Let j be the first job such that $\sum_{k=1}^{j} p_k > T_{\text{LB}} \cdot s_1$. Assign to machine 1 jobs $1, \ldots, j-1$, plus a fraction of j in order to equate $l_1 = T_{\text{LB}} \cdot s_1$. Continue recursively with the unassigned fractions of jobs and with machines $2, \ldots, m$.

Lemma 12.5 *There is enough space to fractionally assign all jobs, and if job j is fractionally assigned to machine i then $p_j/s_i \leq T_{\text{LB}}$.*

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

Lemma 12.6 *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 $l'_i \leq l_i$. Let T'_{LB} denote the new estimate of the optimal makespan. We first claim that $T'_{\text{LB}} \leq \alpha \cdot T_{\text{LB}}$. For an instance $s''_1, \ldots, s''_m$ such that $s''_l = \alpha \cdot s_l$ for all machines l we have that $T''_{\text{LB}} = \alpha \cdot T_{\text{LB}}$ since both terms in the max expression of T_j were multiplied by α. Since $s'_l \leq s_l$ for all l we have that $T'_{\text{LB}} \leq T''_{\text{LB}}$. Now, if $l_i = T_{\text{LB}} \cdot s_i$, i.e. i was full, then $l'_i \leq T'_{\text{LB}} \cdot s'_i \leq T_{\text{LB}} \cdot s_i = l_i$. Otherwise $l_i < T_{\text{LB}} \cdot s_i$, hence i is the last nonempty machine. Since $T'_{\text{LB}} \geq T_{\text{LB}}$, 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). We offer two types of rounding, a randomized rounding and a deterministic one. The former is simpler, and results in a better approximation ratio, but uses the weaker solution concept of truthfulness in expectation. The latter is slightly more involved, and uses deterministic truthfulness, but results in an inferior approximation ratio.

Definition 12.7 (A randomized rounding) 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 12.8 *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 12.5 we have that $\max\{p_j, p_k\} \leq T_{\text{LB}}$ and since T_{LB} is not larger than the optimal 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 probability 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 12.6. □

An integral deterministic algorithm. To be accurate, what follows is not exactly a rounding of the fractional assignment we obtained above, but a similar-in-spirit deterministic assignment. We set virtual speeds, where the fastest machine is set to be slightly faster, and the others are set to be slightly slower, we find a fractional assignment according to these virtual speeds, and then use the "natural" rounding of placing each job fully on the first machine it is fractionally assigned to. With these virtual speeds, the rounding that previously failed to be monotone, now succeeds:

Definition 12.9 (A deterministic algorithm) Given the bids $s_1, \ldots, s_m$, perform:

(i) Set new (virtual) speeds $d_1, \ldots, d_m$, as follows. Let $d_1 = \frac{8}{5}s_1$, and for $i \geq 2$, let d_i be the the closest value of the "breakpoints" $\frac{s_1}{2.5^i}$ (for $i = 1, 2, \ldots$) such that $d_i \leq s_i$.

(ii) Compute T_{LB} according to the virtual speeds, i.e. $T_{\mathrm{LB}} = T_{\mathrm{LB}}(d_i, d_{-i})$.

(iii) Assign jobs to machines, starting from the largest job and the fastest machine. Move to the next machine when the current machine, i, holds jobs with total processing time larger or equal to $T_{\mathrm{LB}} \cdot d_i$.

Note that if the fastest machine changes its speed, then *all* the d_i's may change. Also note that step 3 manages to assign all jobs, since what we are doing is exactly the deterministic natural rounding described above for the fractional assignment, *using the d_i's instead of the s_i's*. As we shall see, this crucial difference enables monotonicity, in the cost of a certain loss in the approximation.

To exactly see the approximation loss, first note that $T_{\mathrm{LB}}(d) \leq 2.5T_{\mathrm{LB}}(s)$, since speeds are made slower by at most this factor. For the fastest machine, since s_1 is lower than d_1, the actual load up to $T_{\mathrm{LB}}(d)$ may be $1.6T_{\mathrm{LB}}(d) \leq 4T_{\mathrm{LB}}(s)$. As we may integrally place on machine 1 one job that is partially assigned also to machine 2, observe (i) that $d_1 \geq 4d_2$, and (ii) by the fractional rules the added job has load at most $T_{\mathrm{LB}}(d)d_2$. Thus get that the load on machine 1 is at most $\frac{5}{4}1.6T_{\mathrm{LB}}(d) \leq 5T_{\mathrm{LB}}(s)$. For any other machine, $d_i \leq s_i$, and so after we integrally place the one extra partial job the load can be at most $2T_{\mathrm{LB}}(d)d_i \leq 2 \cdot 2.5T_{\mathrm{LB}}(s)s_i = 5T_{\mathrm{LB}}(s)s_i$. Since $T_{\mathrm{LB}}(s)$ lower bounds the optimal makespan for s the approximation follows.

To understand why monotonicity holds, we first need few observations that easily follow from our knowledge on the fractional assignment.

For any $i > 1$ and $\beta < d_i$, $T_{\mathrm{LB}}(\beta, d_{-i}) \leq \frac{5}{4}T_{\mathrm{LB}}(d_i, d_{-i})$. Consider the following modification to the fractional assignment for (d_i, d_{-i}): machine i does not get any job, and each machine $1 \leq i' < i$ gets the jobs that were previously assigned to machine $i' + 1$. Since i' is faster than $i' + 1$, any machine $2 \leq i' < i$ does not cross the $T_{\mathrm{LB}}(d_i, d_{-i})$ limit. As for machine 1, note that it is always the case that $d_1 \geq 4d_2$, hence the new load on machine 1 is at most $\frac{5}{4}T_{\mathrm{LB}}(d_i, d_{-i})$.

If a machine $i > 1$ slows down then the total work assigned to the faster machines does not decrease, which follows immediately from the fact that $T_{\mathrm{LB}}(d_i', d_{-i}) \geq T_{\mathrm{LB}}(d_i, d_{-i})$, for $d_i' \geq d_i$.

If the fastest machine slows down, yet remains the fastest, then its assigned work does not increase. Let $s_1' = c \cdot s_1$ for some $c < 1$. Therefore all breakpoints shift by a factor of c. If no speed s_i moves to a new breakpoint then all d's move by a factor of c, the resulting T_{LB} will therefore also move by a factor of c, meaning that machine 1 will get the same set of jobs as before. If additionally some s_i's move to a new breakpoint this implies that the respective d_i's decrease, and by the monotonicity of T_{LB} it also decreases, which means that machine 1 will not get more work.

Lemma 12.10 *The deterministic algorithm is monotone.*

PROOF Suppose that machine i slows down from s_i to $s_i' < s_i$. We need to show that it does not get more work. Assume that the vector d has indeed changed because of i's change.

If i is the fastest machine and it remains the fastest then the above observation is what we need. If the fastest machine changes to i', then we add an artificial breakpoint to the slowdown decrease, where i and i''s speeds are identical, and the title of the "fastest machine" moves from i to i'. Note that the same threshold, T, is computed when the title goes from i to i'. i's work when it is the "fastest machine" is at least $\frac{8}{5}s_i \cdot T$, while i's work when i' is the fastest is at most $2\frac{s_1}{2.5}T < \frac{8}{5}s_i \cdot T$, hence decreases.

If i is not the fastest, but still full, then $d_i' < d_i$ (since the breakpoints remain fixed), and therefore $T_{\mathrm{LB}}(d_i', d_{-i}) \leq \frac{5}{4}T_{\mathrm{LB}}(d_i, d_{-i})$. With s_i, $i's$ work is at least $T \cdot d_i$ (where $T = T_{\mathrm{LB}}(d_i, d_{-i})$), and with s_i' its work is at most $2 \cdot \frac{5}{4}T\frac{d_i}{2.5} = T \cdot d_i$, hence i's load does not increase.

Finally, note that if i's is not full then by the third observation, since the work of the previous machines does not decrease, then i's work does not increase. □

By the above arguments we immediately get the following theorem.

Theorem 12.11 *There exists a truthful deterministic mechanism for scheduling related machines, that approximates the makespan by a factor of* 5.

A note about price computation is in place. A polynomial-time *mechanism* must compute the prices in polynomial time. To compute the prices for both the randomized and the deterministic mechanisms, we need to integrate over the load function of a player, fixing the others' speeds. In both cases 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 can see that price computation is polynomial-time.

Without the monotonicity requirement, a PTAS for related machines exists. The question whether one can incorporate truthfulness is still open.

Open Question Does there exist a *truthful* PTAS for related machines?

The technical discussion of this section aims to demonstrate that, for single-dimensional domains, the algorithmic implications of the game-theoretic requirement are "manageable," and leave ample flexibility for the algorithmic designer. *Multi*-dimensionality, on the other hand, does not exhibit this easy structure, and the rest of this chapter is concerned with exactly this issue.

12.3 Multidimensional Domains: Combinatorial Auctions

As opposed to single-dimensional domains, the monotonicity conditions that characterize implementability in multidimensional domains are far more complex (see the discussion in Chapter 9), hence designing implementable approximation algorithms is harder. As discussed in the Introduction, this chapter examines three aspects of this issue, and in this section we will utilize randomness to overcome the difficulties of implementability in multidimensional domains. We study this for the representative and central problem domain of Combinatorial Auctions.

Combinatorial Auctions (CAs) are a central model with theoretical importance and practical relevance. 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 CAs, providing a comprehensive discussion on the model and its various computational aspects. Our focus here is different: 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 $(S_1, \ldots, S_n)$ that maximizes $\sum_i v_i(S_i)$.

Since a general valuation has size exponential in n and m, the representation issue must be taken into account. Chapter 11 examines two models. In the *bidding languages* model, the bid of a player represents his valuation in a concise way. For this model it is NP-hard to approximate the social welfare within a ratio 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 ratio of $\Omega(m^{1/2-\epsilon})$ for any $\epsilon > 0$. These bounds are tight, as there exists 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 if we attempt to use classic techniques, and our aim is to develop a new technique that will combine the two desirable aspects of efficient computation and incentive compatibility.

We describe a rather general LP-based 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:

$$\begin{aligned} \max \quad & \sum_{i,S\neq\emptyset} x_{i,S}\cdot v_i(S) & & \text{(CA-P)} \\ \text{subject to} \quad & \sum_{S\neq\emptyset} x_{i,S} \leq 1 \quad \text{for each player } i & & (12.4) \\ & \sum_{i} \sum_{S:j\in S} x_{i,S} \leq 1 \quad \text{for each item } j & & (12.5) \\ & x_{i,S} \geq 0 \quad \forall i, S \neq \emptyset. \end{aligned}$$

By constraint 12.4, a player receives at most one integral subset, and constraint 12.5 ensures that each item is not overallocated. 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 (e.g., 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 nonzero $x_{i,S}$ 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 12.12 *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 12.13 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 12.14 (The decomposition lemma) *Suppose that 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 12.15 (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 nonnegative utility, regardless of the randomized choice)[1]: 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 12.16 *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., $E_{\lambda_l}[p_i^R(v)] = p_i^F(v)/c$:

$$\begin{aligned} E_{\{\lambda_l\}_{l\in\mathcal{I}}}\big[p_i^R(v)\big] &= \sum_{l\in\mathcal{I}} \lambda_l \cdot [v_i(x^l)/v_i(x^*)] \cdot p_i^F(v) \\ &= \big[p_i^F(v)/v_i(x^*)\big] \cdot \sum_{l\in\mathcal{I}} \lambda_l \cdot v_i(x^l) \\ &= \big[p_i^F(v)/v_i(x^*)\big] \cdot v_i(x^*/c) = p_i^F(v)/c \end{aligned} \tag{12.6}$$

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

$$v_i(x^*) - p_i^F(v_i, v_{-i}) \geq v_i(z^*) - p_i^F(v_i', v_{-i}) \tag{12.7}$$

By 12.6 and by the decomposition, dividing 12.7 by c yields

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

[1] See Chapter 9 for definitions and a discussion on randomized mechanisms.

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 the weaker solution concept of *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 strategy. 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.

$$\min \sum_{l\in\mathcal{I}} \lambda_l \qquad \text{(P)}$$
s.t.
$$\sum_l \lambda_l x^l_{i,S} = \frac{x_{i,S}}{c} \quad \forall (i,S)\in E \qquad (12.8)$$
$$\sum_l \lambda_l \geq 1$$
$$\lambda_l \geq 0 \qquad \forall l \in \mathcal{I}$$

$$\max \frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \qquad \text{(D)}$$
s.t.
$$\sum_{(i,S)\in E} x^l_{i,S} w_{i,S} + z \leq 1 \;\forall l \in \mathcal{I} \qquad (12.9)$$
$$z \geq 0$$
$$w_{i,S} \text{ unconstrained} \qquad \forall (i,S)\in E.$$

Constraints 12.8 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 12.8 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 12.17 *If w, z is feasible for D then $\frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \leq 1$. 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 $\sum_{(i,S)\in E} x^l_{i,S} w_{i,S} \geq \frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} > 1 - z$, 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 12.9 is violated (for x^l). □

Corollary 12.18 *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 12.17 it is at most 1. To solve P, we first solve D with the following separation oracle: given w, z, if $\frac{1}{c}\sum_{(i,S)\in E} x_{i,S} w_{i,S} + z \leq 1$, 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. □

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 12.19 *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. □

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 12.20 (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 12.21 *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 $Y_i = \{\ (i', S) \in E \mid y_{i',S} > 0$ and (i', S) was removed from E when (i, S_i) 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

$$\begin{aligned}\sum_{(i',S)\in Y_i} y_{i',S} w_{i',S} &= \sum_{(i',S)\in Y_i} y_{i',S} \frac{w_{i',S}}{\sqrt{|S|}}\sqrt{|S|} \\ &\leq \frac{w_{i,S_i}}{\sqrt{|S_i|}} \sum_{(i',S)\in Y_i} y_{i',S}\cdot\sqrt{|S|} \\ &\leq \frac{w_{i,S_i}}{\sqrt{|S_i|}} \sqrt{\left(\sum_{(i',S)\in Y_i} y_{i',S}\right)\left(\sum_{(i',S)\in Y_i} y_{i',S}\cdot|S|\right)} \qquad (12.10)\end{aligned}$$

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:

$$\sum_{(i',S)\in Y_i} y_{i',S} \leq \sum_{j\in S_i}\ \sum_{(i',S)\in Y_i, j\in S} y_{i',S} + \sum_{(i,S)\in Y_i} y_{i,S} \leq \sum_{j\in S_i} 1 + 1 \leq |S_i| + 1 \qquad (12.11)$$

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,

$$\sum_{(i',S)\in Y_i} y_{i',S}\cdot|S| \leq \sum_{j\in\Omega}\ \sum_{(i',S)\in Y_i, j\in S} y_{i',S} \leq m \qquad (12.12)$$

Combining 12.10, 12.11, and 12.12, we get what we need:

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

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

Theorem 12.22 *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 multiunit 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 sub-modular 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.

12.3.1 A General Overview of Truthful Combinatorial Auctions

The search for truthful CAs is an active field of research. Roughly speaking, two techniques have 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 $O(\sqrt{m})$-approximation for subadditive valuations (Dobzinski et al., 2005), an $O(\frac{m}{\sqrt{\log m}})$-approximation for general valuations (Holzman et al., 2004), and a 2-approximation. when all items are identical ("multi-unit auctions") (Dobzinski and Nisan, 2006). 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 $O(\sqrt{m})$-approximation. for general valuations, and an $O(\log^2 m)$ for XOS valuations (Dobzinski et al., 2006). The truthfulness here is "universal," i.e., for any coin toss – a stronger notion than truthfulness in expectation. Bartal et al. (2003) use a similar idea to obtain a truthful and *deterministic* $O(B \cdot m^{\frac{1}{B-2}})$-approximation for multiunit 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 (Dobzinski and Nisan, 2006 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 that are NP-hard and yet constant algorithmic approximations are known? For example, does there exist a truthful constant-factor approximation for CAs with submodular valuations?

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. Lavi et al. (2003) have initiated an investigation for the first part of the argument, obtaining only partial results. Dobzinski and Nisan (2006) 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?

This section was devoted to welfare maximization. Revenue maximization is another important goal for CA design. The mechanism of Bartal et al. (2003) obtains the same guarantees with respect to the optimal revenue. More tight results for multi-unit auctions with budget constrained players are given by Borgs et al. (2005), and for unlimited-supply CAs by Balcan et al. (2005). It should be noted that these are preliminary results for special cases; this issue is still quite unexplored.

12.4 Impossibilities of Dominant Strategy Implementability

In the previous sections we saw an interesting contrast between deterministic and randomized truthfulness, where the key difference seems to be the dimensionality of the domain. We now ask whether the source of this difficulty can be rigorously identified and characterized. What exactly do we mean by an "impossibility," especially since we know that VCG mechanisms *are* possible, in every domain? Well, we mean that nothing *besides* VCG is possible. Such a situation should be viewed as an impossibility, since (i) many times VCG is computationally intractable (as we saw for CAs), and (ii) many times we seek goals different from welfare maximization (as we saw for scheduling domains). The monotonicity characterizations of Chapter 9 almost readily provide few easy impossibilities for some special domains (see the exercises at the end of this chapter), and in this section we will study a more fundamental case.

To formalize our exact question, it will be convenient to use the abstract social choice setting introduced in Chapter 9: 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 $f : V_1 \times \cdots \times V_n \rightarrow A$ (where w.l.o.g. assume that $f : V \rightarrow A$ is onto A). From chapter 9 we know that VCG implements welfare maximization, for any domain, and that affine maximizers are also always implementable.

Definition 12.23 (Affine maximizer) f is an "affine maximizer" if there exist weights $k_1, \ldots, k_n$ and $\{C_x\}_{x \in A}$ such that, for all $v \in V$,

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

The fundamental question is what other function forms are implementable. This question has remained mostly unexplored, with few exceptions. In particular, if the domain is unrestricted, the answer is sharp.

Theorem 12.24 *Suppose $|A| \geq 3$ and $V_i = \Re^A$ for all i. Then f is dominant-strategy implementable iff it is an affine maximizer.*

We will prove here a slightly easier version of the sufficiency 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 12.25 (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 12.26 *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. Recall that Chapter 9 portrayed the strong connection between implementability and certain monotonicity properties. The monotonicity conditions that we consider here are stronger, and are not necessary for all domains. However, for an unrestricted domain, their importance will soon become clear.

Definition 12.27 (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 12.28 *Any implementable function f, on any domain, satisfies PAD.*

PROOF Let $v^i = (v'_1, \ldots, v'_i, v_{i+1}, \ldots, v_n)$, 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, weak monotonicity can be generalized as follows.

Definition 12.29 (Generalized-WMON) For every $v, v' \in V$ with $f(v) = x$ and $f(v') = y$ there exists a player i such that $v'_i(y) - v_i(y) \geq v'_i(x) - v_i(x)$.

With weak monotonicity, 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 12.30 *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 that $f(v) = y$. Fix $\Delta \in \Re^n$ such that $v'(x) - v'(y) = v(x) - v(y) - \Delta$, and define v'':

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

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

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

$$P(x, y) = \{\alpha \in \Re^n \mid \exists v \in V:\ v(x) - v(y) = \alpha,\ f(v) = x\ \}. \qquad (12.13)$$

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 12.31 *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. □

Claim 12.32 *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 12.33 *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 12.32 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 12.34 *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.[2] By Claim 12.32, $\alpha + \beta \in P$. 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 12.31, $-\alpha/2 \in P$. Then $\alpha/2 = \alpha + (-\alpha/2) \in P$, a contradiction. □

We now conclude the proof of Theorem 12.26. 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 that $f(v) = x$ 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.

We have just seen a unique example, demonstrating that there exists a domain for which affine maximizers are the only possibility. However, our natural focus is on *restricted* domains, as most of the computational models that we consider do have some structure (e.g., the two domains we have considered in this chapter). Unfortunately, clear-cut impossibilities for such domains are not known.

Open Question Characterize the class of domains for which affine maximizers are the only implementable functions.

Even this question does not capture the entire picture, as, for example, it is known that there exists an implementable but not an affine-maximizer CA.[3] Nevertheless, there do seem to be some inherent difficulties in designing truthful and computationally-efficient CAs.[4] The less formal open question therefore searches for the fundamental issues that cause the clash. Obviously, these are related to the monotonicity conditions, but an exact quantification of this is still unknown.

[2] 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.

[3] See Lavi et al. (2003).

[4] Note that we have in mind deterministic CAs.

12.5 Alternative Solution Concepts

In light of the conclusions of the previous section, a natural way to advance would be to reexamine the *solution concept* that we are using. In Section 12.3 we saw that randomization certainly helps, but also carries with it some disadvantages. However, in some cases randomization is not known to help, and additionally sometimes we want to stick to deterministic mechanisms. What other solution concepts that fit the worst-case way of thinking in CS can we use?

One simple thought is that algorithm designers do not care so much about actually reaching an equilibrium point – our major concern is to guarantee the optimality of the solution, taking into account the strategic behavior of the players. One way of doing this is to reach a good equilibrium point. But there is no reason why we should not allow the mechanism designer to "leave in" several acceptable strategic choices for the players, and to require the approximation to be achieved *in each of these choices*.

As a first attempt, one is tempted to simply let the players try and improve the basic result by allowing them to lie. However, this can cause unexpected dynamics, as each player chooses her lies under some assumptions about the lies of the others, etc. etc. We wish to avoid such an unpredictable situation, and we insist on using rigorous game theoretic reasoning to explain exactly why the outcome will be satisfactory. The following definition captures the initial intuition, without falling to such pitfalls:

Definition 12.35 (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 a 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 he will 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 the notion of 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.

In this section we illustrate how to use such a solution concept to design CAs for a special class of "single-value" players. The resulting auction has another interesting feature: while most mechanisms we have seen so far are direct revelation, in practice indirect mechanisms, and especially ascending auctions (players compete by raising prices and winners pay their last bid) are much preferred. The following result is an attempt to handle this issue as well.

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, $v_i(s) \in \{0, \bar{v}_i\}$. That is, 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 subprocedure, which will eventually convert algorithms to a mechanism, 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 "tentative winning bundle" s_i^j for every i. In each iteration, the subprocedure is invoked to update the set of winners to W_{j+1} and the winning bundles to s^{j+1}. Every active nonwinner then chooses to double his bid (v_i^j) or to permanently retire. This is iterated until all nonwinners retire.

Definition 12.36 (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. $(W_{j+1}, s^{j+1}) \leftarrow \text{PROC}(v^j, s^j, W_j)$.
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 $j \rightarrow j+1$, 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 12.37 (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:** $\forall i$, $s_i^{j+1} \subseteq s_i^j$.

In words, the pareto property implies that the set of winners that PROC outputs is maximal, i.e., that any loser that has not retired desires a bundle that intersects some winner's bundle. The shrinking-sets property says that a player's new tentative bundle must be a subset of the old tentative bundle.

A "reasonable" player will not increase v_i^j above $\bar{v}_i$; otherwise, his utility will be nonpositive (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 12.38 *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 analysis. To warm up and to collect basic insights, we first 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 lie only about her value). This allows for a simple analysis: the wrapper converts *any* given c-approximation. 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 12.39 (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 12.40 (The KSM-PROC) Given a c-approximation. A for KSM players, KSM-PROC invokes A with s^j (the desired bundles) and v^j (player values). Then, it postprocesses the output to verify pareto and improvement.

Proposition 12.41 *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

$$R_j(\vec{v}, \vec{S}) = \{ (v_i, S_i|_{s_i^j}) | i \text{ retired at iteration } j \}, \tag{12.14}$$

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 12.42 (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 12.43 *Given a c-approximation A for single minded players, KSM-PROC is a c-local-approximation for the set D of dominant strategies.*

PROOF The algorithmic approximation property follows since A outputs a c-approximation outcome. The value bounds property is exactly Proposition 12.41. □

We next translate local approximation to global approximation (this is valid also for the single-value case).

Claim 12.44 *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, $OPT(R_j(\bar{v}, \bar{S})) \leq 2 \cdot OPT(R_j(v^j, \bar{S}))$. We have (i) $OPT(R_j(v^j, \bar{S})) \leq c \cdot \sum_{i \in W_j} v_i^j$ by algorithmic approximation, (ii) $\sum_{i \in W_j} v_i^j \leq \sum_{i \in W_{j+1}} v_i^{j+1}$ by improvement, and (iii) $v_i^J \leq \bar{v}_i$ (by the value bounds), and therefore 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 12.45 *Given any c-approximation. for KSM players, the Wrapper with KSM-PROC implements an* $O(\log(v_{\max}) \cdot c)$ *approximation. in dominant strategies.*

A subprocedure for single-value players. Two assumptions are relaxed: players are now multiminded, and their desired bundles are unknown. Here, we define the following specific subprocedure. 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 12.46 (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 nonwinners 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 a "reasonable" player should 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 is exactly the point where we use the new notion of algorithmic implementation.

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 12.47 *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 none 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 12.44 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 12.48 (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 12.49 *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 12.44. 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 12.50 $OPT(P) \leq J \cdot OPT(\bar{R})$.

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, $s_{i_1}^j \cap s_{i_2}^j = \emptyset$ 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_i^j. 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 nonretiring players belong to W_J. From the above we have $OPT(P \cup \bar{R}) \leq OPT(P) + OPT(\bar{R}) \leq J \cdot OPT(\bar{R}) + OPT(\bar{R}) \leq 4 \cdot J^2 \cdot c \cdot \sum_{i \in W_J} \bar{v}_i^J$. Since s_i^J 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 $OPT(P \cup \bar{R} \cup W_J) \leq 5 \cdot J^2 \cdot \tilde{c} \cdot \sum_{i \in W_J} \bar{v}_i^J$. Since $J \leq 2 \cdot \log(v_{\max}) + 1$ by Lemma 12.38, the lemma follows. □

By all the above, we conclude the following.

Theorem 12.51 *The Wrapper with 1-CA-PROC is an algorithmic implementation of an $O(\log^2(v_{\max}) \cdot c)$-approximation for single-value players.*

This result has demonstrated that if we are less interested in reaching an equilibrium point, but rather in guaranteeing a good-enough outcome, then alternative solution concepts, that are no worse than classic dominant strategies, can be of much help. However, the true power of relaxing dominant strategies to undominated strategies was not formally settled.

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?

12.6 Bibliographic Notes

The connection between classic scheduling and mechanism design was suggested by Nisan and Ronen (2001), that studied unrelated machines and reached mainly impossibilities. Archer and Tardos (2001) studied the case of related machines, and the monotonicity characterization of Section 12.2 is based on their work. Deterministic mechanisms for the problem have been suggested by several works, and the algorithm presented here is by Andelman, Azar, and Sorani (2005). The current best approximation ratio, 3, is given by Kovacs (2005). Section 12.3 is based on the work of Lavi and Swamy (2005). Roberts (1979) characterized dominant strategy implementability for unrestricted domains. The proof given here is based on Lavi, Mu'alem, and Nisan (2004). Generalized-WMON was suggested by Lavi, Mu'alem, and Nisan (2003), which explored the same characterization question for restricted domains in general, and for CAs in particular. Section 12.5 is based on the work of Babaioff, Lavi, and Pavlov (2006). There have been several other suggestions for alternative solution concepts. For example, Kothari et al. (2005) describe an "almost truthful" deterministic FPAS for multiunit auctions, and Lavi and Nisan (2005) define a notion of "Set-Nash" for multi-unit auctions in an online setting, for which they show that deterministic truthfulness obtains significantly lower approximations than Set-Nash implementations.

Bibliography

N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *Proc. of the 22nd Intl. Symp. Theor. Asp. Comp. Sci. (STACS)*, pp. 69–82, 2005.

A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the 42nd Annual Symp. Fdns. of Computer Science*, 2001.

M. Babaioff, R. Lavi, and E. Pavlov. Single-value combinatorial auctions and implementation in undominated strategies. In *Proc. of the 17th Symp. Discrete Algorithms*, 2006.

M. Balcan, A. Blum, J. Hartline, and Y. Mansour. Mechanism design via machine learning. In *Proc. of the 46th Annual Symp. Fdns. of Computer Science*, 2005.

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

C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. In *Proc. of the 6th ACM Conf. Electronic Commerce (ACM-EC)*, 2005.

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

S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proc. of the 37th ACM Symp. Theory of Computing*, 2005.

S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proc. of the 38th ACM Symp. Theory of Computing*, 2006.

R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games Econ. Behav.*, 47:104–123, 2004.

A. Kothari, D. Parkes, and S. Suri. Approximately-strategy proof and tractable multi-unit auctions. *Decis. Support Systems*, 39:105–121, 2005.

A. Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In *Proc. of the 13th Annual Eur. Symp. Algo. (ESA)*, 2005.

R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. of the 44th Annual Symp. Fdns. of Computer Science*, 2003.

R. Lavi, A. Mu'alem, and N. Nisan. Two simplified proofs for Roberts' theorem, 2004. Working paper.

R. Lavi and N. Nisan. Online ascending auctions for gradually expiring items. In *Proc. of the 16th Symp. on Discrete Algorithms*, 2005.

R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. of the 46th Annual Symp. Fdns. of Computer Science*, 2005.

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

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

Exercises

12.1 (Scheduling related machines) Find an implementable algorithm that exactly obtains the optimal makespan, for scheduling on related machines (since this is an NP-hard problem, obviously you may ignore the computational complexity of your algorithm).

12.2 (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 2. Hint: Start with four jobs that have $p_{ij} = 1$ for all i, j.

12.3 A deterministic greedy rounding of the fractional scheduling 12.4 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.

12.4 Prove that 1-CA-PROC of Definition 12.46, and Greedy for multiminded players of Definition 12.20 are not dominant-strategy implementable.

12.5 (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).

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

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

12.8 This exercise aims to complete the characterization of Section 12.4:
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).