Lecture 4

Algorithmic Mechanism Design

This lecture pursues mechanisms that are DSIC, welfare maximizing, and computationally efficient for single-parameter environments that are more complex than those in Lectures 2 and 3. These environments are general enough that the welfare maximization problem is $\mathcal{NP}$-hard, so we consider allocation rules that only maximize the social welfare approximately. There are many techniques for designing such rules, but not all of them yield rules that are monotone in the sense required by Myerson's lemma. This lecture also discusses the revelation principle, the formal justification for our restriction to direct-revelation mechanisms.

Section 4.1 introduces knapsack auctions, which are conceptually simple single-parameter environments in which welfare maximization is a computationally intractable (i.e., $\mathcal{NP}$-hard) problem. Section 4.2 uses knapsack auctions to illustrate some representative results in algorithmic mechanism design, where the goal to design DSIC and polynomial-time mechanisms that guarantee near-optimal welfare. Section 4.3 presents the revelation principle.

4.1 Knapsack Auctions

4.1.1 Problem Definition

Knapsack auctions are another example of single-parameter environments (Section 3.1).

Example 4.1 (Knapsack Auction) In a knapsack auction, each bidder i has a publicly known *size* w_i and a private valuation. The seller has a capacity W. The feasible set X is defined as the 0-1 vectors $(x_1, \ldots, x_n)$ such that $\sum_{i=1}^{n} w_i x_i \leq W$. (As usual, $x_i = 1$ indicates that i is a winning bidder.)

Whenever there is a shared resource with limited capacity, you have a knapsack problem. For instance, each bidder's size could represent the duration of a company's television ad, the valuation its willingness-to-pay for its ad being shown during the Super Bowl, and the seller capacity the length of a commercial break. Other situations that can be modeled with knapsack auctions include bidders who want files stored on a shared server, data streams sent through a shared communication channel, or processes to be executed on a shared supercomputer. A k-unit auction (Example 3.2) corresponds to a knapsack auction with $w_i = 1$ for all i and $W = k$.

Let's try to design an ideal auction using our two-step design paradigm (Section 2.6.4). First, we assume without justification that we receive truthful bids and decide on our allocation rule. Then we pay the piper and devise a payment rule that extends the allocation rule to a DSIC mechanism.

4.1.2 Welfare-Maximizing DSIC Knapsack Auctions

Since ideal auctions are supposed to maximize welfare, the answer to the first step is clear: define the allocation rule by

$$\mathbf{x}(\mathbf{b}) = \text{argmax}_X \sum_{i=1}^{n} b_i x_i. \tag{4.1}$$

That is, the allocation rule solves an instance of the knapsack problem[1] in which the item (i.e., bidder) values are the reported bids $b_1, \ldots, b_n$, and the item sizes are the a priori known sizes $w_1, \ldots, w_n$. By definition, when bidders bid truthfully, this allocation rule maximizes the social welfare. This allocation rule is also monotone in the sense of Definition 3.6; see Exercise 4.1.

4.1.3 Critical Bids

Myerson's lemma (Theorem 3.7, parts (a) and (b)) guarantees the existence of a payment rule $\mathbf{p}$ such that the mechanism $(\mathbf{x}, \mathbf{p})$ is DSIC. This payment rule is easy to understand. Fix a bidder i and

[1] An instance of the knapsack problem is defined by $2n + 1$ positive numbers: item values $v_1, \ldots, v_n$, item sizes $w_1, \ldots, w_n$, and a knapsack capacity W. The goal is to compute the subset of items of maximum total value that has total size at most W.

bids $\mathbf{b}_{-i}$ by the other bidders. Since the allocation rule is monotone and assigns 0 or 1 to every bidder, the allocation curve $x_i(\cdot, \mathbf{b}_{-i})$ is 0 until some breakpoint z, at which point it jumps to 1 (Figure 4.1). Recall the payment formula in (3.5):

$$p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{\ell} z_j \cdot [\text{jump in } x_i(\cdot, \mathbf{b}_{-i}) \text{ at } z_j],$$

where $z_1, \ldots, z_\ell$ are the breakpoints of the allocation function $x_i(\cdot, \mathbf{b}_{-i})$ in the range $[0, b_i]$. Thus, if i bids less than z, she loses and pays 0. If i bids more than z, she pays $z \cdot (1 - 0) = z$. That is, when i wins, she pays her *critical bid*—the infimum of the bids she could make and continue to win (holding the other bids $\mathbf{b}_{-i}$ fixed). This is analogous to the familiar payment rule of a single-item second-price auction.

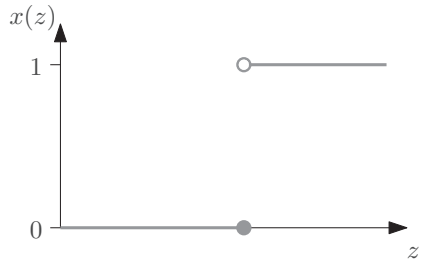

Figure 4.1: A monotone 0-1 allocation rule.

4.1.4 Intractability of Welfare Maximization

Is the mechanism proposed in Section 4.1.2 ideal in the sense of the second-price auction (Theorem 2.4)? Recall this means that the mechanism:

(1) is DSIC;

(2) is welfare maximizing, assuming truthful bids; and

(3) runs in time polynomial in the input size, which is the number of bits needed to represent all of the relevant numbers (bids, sizes, and the capacity).[2]

[2] More precisely, the running time of the mechanism on inputs of size s should be at most cs^d, where c and d are constants independent of s.

The answer is *no*. The reason is that the knapsack problem is $\mathcal{NP}$-hard. This means that there is no polynomial-time implementation of the allocation rule in (4.1), unless $\mathcal{P} = \mathcal{NP}$.[3] Thus, properties (2) and (3) are incompatible.

The fact that there is no ideal knapsack auction (assuming $\mathcal{P} \neq \mathcal{NP}$) motivates relaxing at least one of our three goals. But which one? First, note that relaxing the DSIC condition will not help at all, since it is the second and third properties that conflict.

One sensible approach, which won't get much airtime in this course, is to relax the third constraint. This is particularly attractive for knapsack auctions, since the allocation rule (4.1) can be implemented in pseudopolynomial time using dynamic programming.[4] More generally in mechanism design, if your instances are small or structured enough and you have enough time and computing power to implement optimal welfare maximization, by all means do it. The resulting allocation rule is monotone and can be extended to a DSIC mechanism (Exercise 4.1).[5]

For the rest of this lecture, we'll compromise on the second goal—begrudgingly accepting near-optimal welfare maximization in exchange for computational efficiency and without losing DSIC.

4.2 Algorithmic Mechanism Design

Algorithmic mechanism design is one of the first and most well-studied branches of algorithmic game theory, and this section presents a representative result from the field.

The dominant paradigm in algorithmic mechanism design is to relax the second requirement of ideal auctions (welfare maximization) as little as possible, subject to the first (DSIC) and third (polynomial-time) requirements. For single-parameter environments, Myerson's

[3] $\mathcal{P}$ and $\mathcal{NP}$ denote the sets of problems that can be solved in polynomial time and for which a correct solution can be verified in polynomial time, respectively. $\mathcal{NP}$ can only be larger than $\mathcal{P}$, and $\mathcal{P} \neq \mathcal{NP}$ is an unproven but widely-accepted hypothesis about computation.

[4] That is, if either the bids or the sizes require only a small number of bits to describe, then the problem can be solved efficiently. See any undergraduate algorithms textbook for details.

[5] Don't forget that the payments also need to be computed, and this generally requires solving n more welfare maximization problems (one per agent). See also Exercise 4.3.

lemma (Theorem 3.7) reduces this task to the design of a polynomial-time and monotone allocation rule that comes as close as possible to maximizing the social welfare.

4.2.1 The Best-Case Scenario: DSIC for Free

One reason there has been so much progress in algorithmic mechanism design over the past 15 years is its strong resemblance to the mature field of *approximation algorithms*. The primary goal in approximation algorithms is to design polynomial-time algorithms for $\mathcal{NP}$-hard optimization problems that are as close to optimal as possible. Algorithmic mechanism design has exactly the same goal, except that the algorithms must additionally obey a monotonicity constraint. The incentive constraints of the mechanism design goal are neatly compiled into a relatively intuitive extra constraint on the allocation rule, and so algorithmic mechanism design reduces to algorithm design in an oddly restricted "computational model."

The design space of polynomial-time DSIC mechanisms is only smaller than that of polynomial-time approximation algorithms. The best-case scenario is that the extra DSIC (equivalently, monotonicity) constraint causes no additional welfare loss, beyond the loss we already have to suffer from the polynomial-time requirement. We've been spoiled so far, since exact welfare maximization automatically yields a monotone allocation rule (see Exercise 4.1). Does an analogous fact hold for *approximate* welfare maximization?

4.2.2 Knapsack Auctions Revisited

To explore the preceding question in a concrete setting, let's return to knapsack auctions. There are several heuristics for the knapsack problem that have good worst-case performance guarantees. For example, consider the following allocation rule, which given bids $\mathbf{b}$ chooses a feasible set—a set S of winners with total size $\sum_{i \in S} w_i$ at most the capacity W—via a simple greedy algorithm. Since it's harmless to remove bidders i with $w_i > W$, we can assume that $w_i \leq W$ for every i.

A Greedy Knapsack Heuristic

1. Sort and re-index the bidders so that
$$\frac{b_1}{w_1} \geq \frac{b_2}{w_2} \geq \cdots \geq \frac{b_n}{w_n}.$$[6]
2. Pick winners in this order until one doesn't fit, and then halt.[7]
3. Return either the solution from the previous step or the highest bidder, whichever has larger social welfare.[8]

This greedy algorithm is a $\frac{1}{2}$-approximation algorithm for the knapsack problem, meaning that for every instance of the knapsack problem, the algorithm returns a feasible solution with total value at least $\frac{1}{2}$ times the maximum possible. This fact implies the following guarantee.

Theorem 4.2 (Knapsack Approximation Guarantee) *Assuming truthful bids, the social welfare achieved by the greedy allocation rule is at least 50% of the maximum social welfare.*

Proof (sketch): Consider truthful bids $v_1, \ldots, v_n$, known sizes w_1, $\ldots$, w_n, and a capacity W. Suppose, as a thought experiment, we make the problem easier by allowing bidders to be chosen *fractionally*, with the value prorated accordingly. For example, if 70% of a bidder with value 10 is chosen, then it contributes 7 to the welfare. Here is a greedy algorithm for this "fractional knapsack problem:" sort the bidders as in step (1) above, and pick winners in this order until the entire capacity is fully used (picking the final winner fractionally,

[6] Intuitively, what makes a bidder attractive is a high bid and a small size. This heuristic trades these two properties off by ordering bidders by "bang-per-buck"—the value contributed per unit of capacity used.

[7] Alternatively, continue to follow the sorted order, packing any further bidders that fit. This modified heuristic is only better than the original.

[8] The motivation for this step is that the solution produced by the second step can be highly suboptimal if there is a very valuable and very large bidder. In lieu of considering only the highest bidder, this step can also sort the bidders in nondecreasing bid order and pack them greedily. This modification can only improve the heuristic.

as needed). A straightforward exchange argument proves that this algorithm maximizes the welfare over all feasible solutions to the fractional knapsack problem (Exercise 4.4).

In the optimal fractional solution, suppose that the first k bidders in the sorted order win and that the $(k+1)$th bidder fractionally wins. The welfare achieved by steps (1) and (2) in the greedy allocation rule is exactly the total value of the first k bidders. The welfare of the solution consisting only the highest bidder is at least the fractional value of the $(k+1)$th bidder. The better of these two solutions is at least half of the welfare of the optimal fractional solution, and thus at least half the welfare of an optimal (non-fractional) solution to the original problem. ■

The greedy allocation rule above is also monotone (Exercise 4.5). Using Myerson's lemma (Theorem 3.7), we can extend it to a DSIC mechanism that runs in polynomial time and, assuming truthful bids, achieves social welfare at least 50% of the maximum possible.[9]

You may have been lulled into complacency, thinking that every reasonable allocation rule is monotone. The only non-monotone rule that we've seen is the "second-highest bidder wins" rule for single-item auctions (Section 3.3), which we don't care about anyways. Consider yourself warned.

Warning

Natural allocation rules are not always monotone.

For example, for every $\epsilon > 0$, there is a $(1-\epsilon)$-approximation algorithm for the knapsack problem that runs in time polynomial in the input and $\frac{1}{\epsilon}$—a "fully polynomial-time approximation scheme (FPTAS)" (see Problem 4.2). The rule induced by the standard implementation of this algorithm is *not* monotone, although it can be tweaked to restore monotonicity without degrading the approximation guarantee (again, see Problem 4.2). This is characteristic of work in algorithmic mechanism design: for an $\mathcal{NP}$-hard optimization problem of interest, check if the state-of-the-art approximation

[9] The greedy allocation rule is even better under additional assumptions. For example, if $w_i \leq \alpha W$ for every bidder i, with $\alpha \in (0, \frac{1}{2}]$, then the approximation guarantee improves to $1-\alpha$, even if the third step of the algorithm is omitted.

algorithm directly leads to a DSIC mechanism. If not, tweak it or design a new approximation algorithm that does, hopefully without degrading the approximation guarantee.

4.3 The Revelation Principle

4.3.1 DSIC Revisited

To this point, our mechanism design theory has studied only DSIC mechanisms. We reiterate that there are good reasons to strive for a DSIC guarantee. First, it is easy for a participant to figure out what to do in a DSIC mechanism: just play the obvious dominant strategy and truthfully reveal one's private information. Second, the designer can predict the mechanism's outcome assuming only that participants play their dominant strategies, a relatively weak behavioral assumption. Nevertheless, non-DSIC mechanisms like first-price auctions (Section 2.3) are also important in practice.

Can non-DSIC mechanisms accomplish things that DSIC mechanisms cannot? To answer this question, we need to tease apart two different conditions that are conflated in our DSIC definition (Definition 2.3).[10]

The DSIC Condition

(1) For every valuation profile, the mechanism has a *dominant-strategy equilibrium*—an outcome that results from every participant playing a dominant strategy.

(2) In this dominant-strategy equilibrium, every participant truthfully reports her private information to the mechanism.

There are mechanisms that satisfy (1) but not (2). To give a silly example, imagine a single-item auction in which the seller, given bids $\mathbf{b}$, runs a Vickrey auction on the bids $2\mathbf{b}$. Every bidder's dominant strategy is then to bid half her value.

[10] We'll ignore the "individual rationality" condition in Definition 2.3, which does not matter for the main points of this section.

4.3.2 Justifying Direct Revelation

The revelation principle states that, given requirement (1) in Section 4.3.1, requirement (2) comes for free.

Theorem 4.3 (Revelation Principle for DSIC Mechanisms) *For every mechanism M in which every participant always has a dominant strategy, there is an equivalent direct-revelation DSIC mechanism M'.*

"Equivalence" in Theorem 4.3 means that, for every valuation profile, the outcome (e.g., winners of an auction and selling prices) of M' under direct revelation is identical to that of M when agents play their dominant strategies.

Proof: The proof uses a simulation argument; see Figure 4.2. By assumption, for every participant i and private information v_i that i might have, i has a dominant strategy $s_i(v_i)$ in the given mechanism M.

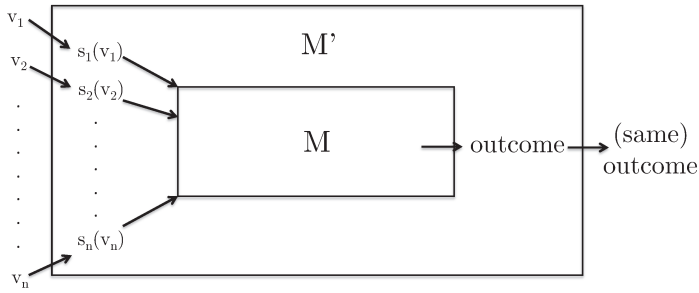

Figure 4.2: Proof of the revelation principle. Construction of the direct-revelation mechanism M', given a mechanism M with dominant strategies.

We next construct a mechanism M', to which participants delegate the responsibility of playing the appropriate dominant strategy. Precisely, the (direct-revelation) mechanism M' accepts bids $b_1, \ldots, b_n$ from the agents. It submits the bids $s_1(b_1), \ldots, s_n(b_n)$ to the mechanism M and chooses the same outcome that M does.

Mechanism M' is DSIC: If a participant i has private information v_i, then submitting a bid other than v_i can only result in M' playing a strategy other than $s_i(v_i)$ in M, which can only decrease i's utility. ■

The point of Theorem 4.3 is that, at least in principle, if you want to design a mechanism with dominant strategies, then you might as well design one in which direct revelation (in auctions, truthful bidding) is a dominant strategy. Thus *truthfulness per se is not important*; what makes DSIC mechanism design difficult is the requirement that a desired outcome is a dominant-strategy equilibrium.

4.3.3 Beyond Dominant-Strategy Equilibria

Can we obtain better mechanisms by relaxing condition (1) from Section 4.3.1? An immediate issue with this idea is that, when agents do not have dominant strategies, we require stronger behavioral assumptions to predict what participants will do and what the mechanism's outcome will be. For example, we can consider a Bayes-Nash equilibrium with respect to a common prior distribution over the private information of the participants (see Problem 5.3) or a Nash equilibrium in a full-information model (similar to Problem 3.1). If we're willing to make such assumptions, can we do better than with DSIC mechanisms?

The answer is "sometimes, yes." For this reason, and because non-DSIC mechanisms are common in practice, it is important to develop mechanism design theory beyond DSIC mechanisms. Remark 5.5 and Problem 5.3 offer a brief glimpse of this theory. A rough rule of thumb is that, for sufficiently simple problems like those we've studied up until now, DSIC mechanisms can do everything that non-DSIC mechanisms can. In more complex problems, however, weakening the DSIC constraint often allows the designer to achieve performance guarantees that are provably impossible for DSIC mechanisms. DSIC and non-DSIC mechanisms are incomparable in such settings—the former enjoy stronger incentive guarantees, the latter better performance guarantees. Which of these is more important depends on the details of the application.

The Upshot

☆ Knapsack auctions model the allocation of a shared resource with limited capacity. Bidders have private valuations and publicly known sizes. In a feasible outcome, the total size of

the winning bidders is at most the resource capacity.

☆ The problem of computing the outcome of a knapsack auction that maximizes social welfare is $\mathcal{NP}$-hard. Thus, if $\mathcal{P} \neq \mathcal{NP}$, there are no ideal knapsack auctions.

☆ The goal in algorithmic mechanism design is to relax the second requirement of ideal auctions (welfare maximization) as little as possible, subject to the first (DSIC) and third (polynomial-time) requirements. In the best-case scenario, there is a polynomial-time DSIC mechanism with an approximate welfare guarantee matching that of state-of-the-art polynomial-time approximation algorithms.

☆ State-of-the-art approximation algorithms for the welfare maximization problem may or may not induce monotone allocation rules.

☆ The revelation principle states that, for every mechanism with a dominant-strategy equilibrium, there is an equivalent mechanism in which direct revelation is a dominant-strategy equilibrium.

☆ In many complex mechanism design problems, non-DSIC mechanisms can achieve performance guarantees that are provably impossible for DSIC mechanisms.

Notes

The origins of algorithmic mechanism design are in Nisan and Ronen (2001) and Lehmann et al. (2002); see Nisan (2015) for a recent survey. Single-parameter environments are studied by Archer and Tardos (2001). Knapsack auctions are introduced in

Mu'Alem and Nisan (2008). The first formulation of the revelation principle appears in Gibbard (1973). Garey and Johnson (1979) give a good exposition of $\mathcal{NP}$-completeness and how to interpret it.

Problem 4.1 is related to Chekuri and Gamzu (2009). The classical FPTAS for the knapsack problem (Problem 4.2) is due to Ibarra and Kim (1975); see the books by Vazirani (2001) and Williamson and Shmoys (2010) for detailed coverage of this and dozens of other state-of-the-art polynomial-time approximation algorithms. The rest of Problem 4.2 is from Briest et al. (2005). Problem 4.3 is from Lehmann et al. (2002).

Exercises

Exercise 4.1 Consider an arbitrary single-parameter environment, with feasible set X. Prove that the welfare-maximizing allocation rule

$$\mathbf{x}(\mathbf{b}) = \text{argmax}_{(x_1,\ldots,x_n)\in X} \sum_{i=1}^{n} b_i x_i \tag{4.2}$$

is monotone in the sense of Definition 3.6.

[Assume that ties are broken in a deterministic and consistent way, such as lexicographically.]

Exercise 4.2 Continuing the previous exercise, restrict now to feasible sets X that contain only 0-1 vectors—that is, each bidder either wins or loses. We can identify each feasible outcome with a "feasible set" of bidders (the winners). Assume that for every bidder i, there is an outcome in which i does not win. Myerson's payment formula (3.5) dictates that a winning bidder pays her "critical bid"—the infimum of the bids at which she would continue to win.

Prove that, when S^* is the set of winning bidders under the allocation rule (4.2) and $i \in S^*$, i's critical bid equals the difference between (1) the maximum social welfare of a feasible set that excludes i; and (2) the social welfare $\sum_{j\in S^*\setminus\{i\}} v_j$ of the bidders other than i in the chosen outcome S^*.

[In this sense, each winning bidder pays her "externality"—the welfare loss she imposes on others.]

Exercise 4.3 Continuing the previous exercise, consider a 0-1 single-parameter environment. Suppose you are given a subroutine that, given bids $\mathbf{b}$, computes the outcome of the welfare-maximizing allocation rule (4.2).

(a) Explain how to implement a welfare-maximizing DSIC mechanism by invoking this subroutine $n + 1$ times, where n is the number of participants.

(b) Conclude that mechanisms that are ideal in the sense of Theorem 2.4 exist for precisely the families of single-parameter environments in which the welfare-maximization problem (given $\mathbf{b}$ as input, compute (4.2)) can be solved in polynomial time.

Exercise 4.4 Prove that the greedy algorithm in the proof of Theorem 4.2 always computes an optimal fractional knapsack solution.

Exercise 4.5 Prove that the three-step greedy knapsack auction allocation rule in Section 4.2.2 is monotone. Does it remain monotone with the two optimizations discussed in the footnotes?

Exercise 4.6 Consider a variant of a knapsack auction in which we have two knapsacks, with known capacities W_1 and W_2. Feasible sets of this single-parameter environment now correspond to subsets S of bidders that can be partitioned into sets S_1 and S_2 satisfying $\sum_{i \in S_j} w_i \leq W_j$ for $j = 1, 2$.

Consider the allocation rule that first uses the single-knapsack greedy allocation rule of Section 4.2.2 to pack the first knapsack, and then uses it again on the remaining bidders to pack the second knapsack. Does this algorithm define a monotone allocation rule? Give either a proof of this fact or an explicit counterexample.

Exercise 4.7 *(H)* The revelation principle (Theorem 4.3) states that (direct-revelation) DSIC mechanisms can simulate all other mechanisms with dominant-strategy equilibria. Critique the revelation principle from a practical perspective. Name a specific situation where you might prefer a non-direct-revelation mechanism with a dominant-strategy equilibrium to the corresponding DSIC mechanism, and explain your reasoning.

Problems

Problem 4.1 Consider a variant of a knapsack auction in which both the valuation v_i and the size w_i of each bidder i are private. A mechanism now receives both bids $\mathbf{b}$ and reported sizes $\mathbf{a}$ from the bidders. An allocation rule $\mathbf{x}(\mathbf{b}, \mathbf{a})$ now specifies the amount of capacity allocated to each bidder, as a function of the bids and reported sizes. Feasibility dictates that $\sum_{i=1}^{n} x_i(\mathbf{b}, \mathbf{a}) \leq W$ for every $\mathbf{b}$ and $\mathbf{a}$, where W is the total capacity of the shared resource. We define the utility of a bidder i as $v_i - p_i(\mathbf{b}, \mathbf{a})$ if she gets her required capacity (i.e., $x_i(\mathbf{b}, \mathbf{a}) \geq w_i$) and as $-p_i(\mathbf{b}, \mathbf{a})$ otherwise. This is not a single-parameter environment.

Consider the following mechanism. Given bids $\mathbf{b}$ and reported sizes $\mathbf{a}$, the mechanism runs the greedy knapsack auction of Section 4.2.2, taking the reported sizes $\mathbf{a}$ at face value, to obtain a subset of winning bidders and prices $\mathbf{p}$. The mechanism concludes by awarding each winning bidder capacity equal to her reported size a_i, at a price of p_i; losing bidders receive and pay nothing. Is this mechanism DSIC? Prove it or give an explicit counterexample.

Problem 4.2 Section 4.2.2 gives an allocation rule for knapsack auctions that is monotone, guarantees at least 50% of the maximum social welfare, and runs in polynomial time. Can we do better?

We first describe a classical fully polynomial-time approximation scheme (FPTAS) for the knapsack problem. The input to the problem is item values $v_1, \ldots, v_n$, item sizes $w_1, \ldots, w_n$, and a knapsack capacity W. For a user-supplied parameter $\epsilon > 0$, we consider the following algorithm $\mathcal{A}_\epsilon$; m is a parameter that will be chosen shortly.

- Round each v_i up to the nearest multiple of m, call it v_i'.
- Divide the v_i''s through by m to obtain integers $\tilde{v}_1, \ldots, \tilde{v}_n$.
- For item values $\tilde{v}_1, \ldots, \tilde{v}_n$, compute the optimal solution using a pseudopolynomial-time algorithm.

  [You can assume that there exists such an algorithm with running time polynomial in n and $\max_{i=1}^{n} \tilde{v}_i$.]

(a) Prove that if we run algorithm $\mathcal{A}_\epsilon$ with the parameter m set to $\epsilon(\max_{i=1}^{n} v_i)/n$, then the running time of the algorithm is polynomial in n and $\frac{1}{\epsilon}$ (independent of the v_i's).

(b) *(H)* Prove that if we run algorithm $\mathcal{A}_\epsilon$ with the parameter m set to $\epsilon(\max_{i=1}^{n} v_i)/n$, then the algorithm outputs a solution with total value at least $1-\epsilon$ times the maximum possible.

(c) Prove that if we run algorithm $\mathcal{A}_\epsilon$ with the parameter m set to a fixed constant, independent of the v_i's, then the algorithm yields a monotone allocation rule.

(d) Prove that if we run algorithm $\mathcal{A}_\epsilon$ with the parameter m set as in (a) and (b), then the algorithm need *not* yield a monotone allocation rule.

(e) *(H)* Give a DSIC mechanism for knapsack auctions that, for a user-specified parameter ϵ and assuming truthful bids, outputs an outcome with social welfare at least $1-\epsilon$ times the maximum possible, in time polynomial in n and $\frac{1}{\epsilon}$.

Problem 4.3 Consider a set M of distinct items. There are n bidders, and each bidder i has a publicly known subset $T_i \subseteq M$ of items that it wants, and a private valuation v_i for getting them. If bidder i is awarded a set S_i of items at a total price of p, then her utility is $v_i x_i - p$, where x_i is 1 if $S_i \supseteq T_i$ and 0 otherwise. This is a single-parameter environment. Since each item can only be awarded to one bidder, a subset W of bidders can all receive their desired subsets simultaneously if and only if if $T_i \cap T_j = \emptyset$ for each distinct $i, j \in W$.

(a) *(H)* Prove that the problem of computing a welfare-maximizing feasible outcome, given the v_i's and T_i's as input, is $\mathcal{NP}$-hard.

(b) Here is a greedy algorithm for the social welfare maximization problem, given bids $\mathbf{b}$ from the bidders.

```
initialize W = ∅ and X = M
sort and re-index the bidders so that
b_1 ≥ b_2 ≥ ··· ≥ b_n
for i = 1, 2, 3, ..., n do
    if T_i ⊆ X then
        remove T_i from X and add i to W
return winning bidders W
```

Does this algorithm define a monotone allocation rule? Prove it or give an explicit counterexample.

(c) *(H)* Prove that if all bidders report truthfully and have sets T_i of cardinality at most d, then the outcome of the allocation rule in (b) has social welfare at least $\frac{1}{d}$ times that of the maximum possible.