Harvard CS 1360, Spring 2026
Instructor: Ariel Procaccia
Scribe: Ethan Tan, Jakob de Raaij

Strategyproof Approximation Algorithms

—

Lecture 9

In the last lecture, we saw that the VCG mechanism is both strategyproof and welfare-maximizing, two highly desirable properties. However, since the VCG mechanism is welfare-maximizing, it needs to compute an allocation that maximizes welfare,

$$f(\mathbf{v}) \in \arg\max_{x \in A} \sum_{i \in N} v_i(x),$$

which in many cases is computationally hard.

Thus, we can only hope to *approximately* maximize welfare in these settings. The standard approach from theoretical computer science would be to use an algorithm for approximately maximizing welfare in the VCG mechanism. However, when doing so, we are no longer guaranteed that the mechanism is strategyproof. Hence, we need to look for *strategyproof approximation algorithms*, which can be used for computationally feasible mechanisms that allow us to retain strategyproofness while approximately maximizing social welfare.

In this lecture, we will consider two problems with strategyproof approximation algorithms: auctioning of a number of items to single-minded bidders and distributing tasks among workers with different skills.

1 Single-Minded Auctions

Definition 1 (Single-Minded Auction)**.** A *single-minded auction* consists of

- a set G of m items (the *goods*),
- a set N of n players, and
- for each player $i \in N$, a *target bundle* $T_i \subseteq G$ and valuation w_i.

The value of player i for receiving goods $S \subseteq G$ is $v_i(S) = w_i$ if $T_i \subseteq S$ and 0 otherwise. That is, player i only cares about receiving *all* items in T_i — if they do, their utility is w_i, else, it is 0.

Example 1 (Single-minded auction)**.** There are 4 items, $\{a, b, c, d\}$, and 4 players with the following target bundles and values. Each column corresponds to one player and a ♥ indicates that the item is in the player's target bundle (so, for example $T_1 = \{a, b\}$ and $w_1 = 5$).

	1	2	3	4
a	♥		♥	
b	♥	♥		
c		♥	♥	♥
d				♥
w_i	5	3	4	2

If we assign player 1 their target bundle $\{a, b\}$, they receive a value of 5. Player 1 also receives a value of 5 if they get additional items, like $\{a, b, c\}$, because they receive their target bundle but have no additional value for the extra item c. If player 1 only receives part or none of their target bundle, like $\{a\}$, they receive a value of 0 because they are missing items from their target.

In this example, we can see that the welfare-maximizing allocation is giving $\{a, b\}$ to player 1 and $\{c, d\}$ to player 4, yielding a social welfare of 7.

Single-minded auctions are an example of a problem where finding the welfare-maximizing allocation is computationally hard.

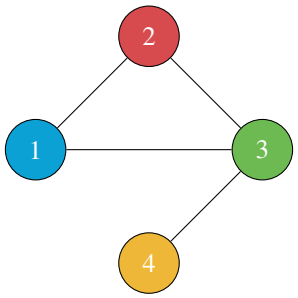
Theorem 1. *Given a single-minded auction and an integer k, deciding whether there exists an allocation of the items with social welfare at least k is **NP**-complete.*

We can show that this problem is **NP**-hard with a reduction from the Maximum Independent Set problem.

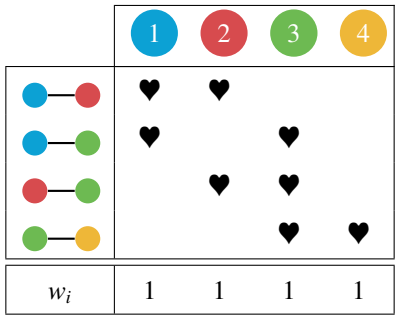
Definition 2 (Maximum Independent Set)**.** Given a graph $\mathcal{G} = (V, E)$ and an integer k, decide whether there exist an *independent set* of size at least k in $\mathcal{G}$; that is, a set $S \subseteq V$ of size $|S| \geq k$ such that no two vertices in S are adjacent (i.e., are connected by an edge).

Proof of Theorem 1. It is immediate that the problem is in **NP**. For **NP**-hardness, let us show that any instance of the Maximum Independent Set problem can be reduced to an instance of the single-minded auction problem.

Given a graph $\mathcal{G} = (V, E)$, construct a single-minded auction with a player for each vertex, $N = V$, and an item for each edge, $G = E$. The target set of player i is the set of all edges incident to their corresponding vertex, i.e., $T_i = \{j : (i, j) \in E\}$. All valuations are 1, i.e., $w_i = 1$. For example, consider applying the reduction to the graph



Our players are the nodes $N = \{1, 2, 3, 4\}$ and the target bundle of any player i is the set of edges connected to node i, giving the target bundles



We claim that there exists an independent set of size k in $\mathcal{G}$ if and only if there exits an allocation with social welfare at least k in the corresponding single-minded auction.

If there exists an independent set of size at least k, we can give all players in the independent set their target bundles: Since no two vertices in the independent set share an edge, no item will appear in more than one of the allocated target bundles. Thus, we find an allocation with social welfare at least k. If there exists an allocation with social welfare at least k, we know that at least k players received their target bundles. Since each item can be in only one allocated target bundle, we know that no two players whose corresponding vertices share an edge received their target bundles. Thus, the vertices corresponding to the (at least k) players that received their target bundle form an independent set of size at least k. □

If there was an efficient algorithm for *finding* a welfare-maximizing allocation, we could also efficiently verify if there exists an allocation with social welfare at least k. However, we know that this is **NP**-hard, so unless **P** = **NP**, no polynomial-time algorithm can always find a welfare-maximizing allocation in single-minded auctions.

2 The Greedy Mechanism

Now that we know that no computationally efficient mechanism can achieve optimal social welfare, we aim for a mechanism that at least approximates social welfare, while also being strategyproof. We consider a greedy mechanism:

Definition 3 (Greedy Mechanism)**.** In the *greedy mechanism* for single-minded auctions, each player i submits a bid (T_i, w_i), where T_i is the target bundle and w_i is the valuation. It uses the following allocation and payment rule:

- Consider the bids (T_i, w_i) in order of decreasing w_i and accept every bid that is feasible when considered. That is, if the target bundle T_i of the current bid is disjoint from all bundles of previously accepted bids, accept it, else reject it. Each player whose bid was accepted receives exactly their target bundle. We call the players whose bid was accepted the *winners*.
- Each winner pays the *critical value*: the smallest alternative valuation w_i' such that their bid would have still been accepted if it was (T_i, w_i'), breaking ties in favor of i. Players whose bid was rejected pay nothing.

Notice the similarity between the payment rule of the greedy mechanism and the Vickrey (second-price) auctions from last lecture. In second-price auctions, analogous to the greedy mechanism, we charge the winner the lowest price they would need to pay while still winning, i.e., the second highest bid.

We'll first show that the greedy mechanism can be computed in polynomial time. It is not hard to see that the allocation rule, due to its greedy nature, is efficient; it remains to show that we can efficiently calculate the critical value for each player. We start by defining the conflict set of each player:

Definition 4 (Conflict Set)**.** Let N_i be the set of winners if player i is not present. The *conflict set* of player i is

$$N_i'(T_i) = \{j \in N_i \mid T_i \cap T_j \neq \emptyset\},$$

the set of all players in N_i whose target bundles overlap with player i's target bundle, i.e., who receive at least one item that player i wants when i is not present.

Lemma 1 (Critical Value Calculation)**.** *The critical value of player i is*

$$w_i^c(T_i) = \max_{j \in N_i'(T_i)} w_j,$$

the maximum valuation of any player in the conflict set of player i.

Proof. For all bids with $w_j > w_i$, the greedy mechanism behaves the same regardless of whether player i participates or not. In particular, if there exists a player in the conflicting set, $j \in N_i'(T_i)$, with bid $w_j > w_i$, we know that the bid of player j will be accepted before the bid of player i is considered. Thus, by the time the bid of player i is considered, at least one item in T_i is already allocated to player j, so the bid of player i is rejected. Thus, we know that player i needs to bid at least $\max_{j \in N_i'(T_i)} w_j$ for their bid to be accepted.

At the same time, we know that bidding $\max_{j \in N_i'(T_i)} w_j$ will always be high enough for their bid to be accepted: When the greedy mechanism considers the bid of player i, $w_i \geq \max_{j \in N_i'(T_i)} w_j$, we know that so far only bids with $w_j > w_i$ have been considered. In particular, all accepted bids so far were from players in $N_i \setminus N_i'(T_i)$, so all items in T_i are still available and will thus be allocated to i. We can conclude that $\max_{j \in N_i'(T_i)} w_j$ is the lowest bid for player i that makes them a winner.

To illustrate this, consider the following example with 6 items and 6 players, where player i (whose critical value we want to determine) is at the bottom of the table.

T_i	Selected?	w_i	
$\{a, b\}$	✓	10	
$\{a, c\}$	✗	8	
$\{d, c\}$	✓	7	$\in N_i'$
$\{d, a\}$	✗	6	
$\{e, f\}$	✓	4	$\in N_i'$
$\{c, f\}$	?	12	

Player i has target bundle $\{c, f\}$. Currently, the greedy mechanism allocates to the first, third, and fifth player. Out of those, the third and fifth player share an element in the bundle with player i, making up N_i'. To be allocated their target bundle, player i thus has to bid at least as much as those players, in particular, their critical value is $\max\{7, 4\} = 7$. □

Using Lemma 1, we can efficiently calculate the payment rule for any player i: We simulate the auction with player i missing to obtain N_i, from which we can efficiently get $N_i'(T_i)$ and w_i^c.

Now that we know that the greedy mechanism is efficient, let's consider its other properties. We will first show that it is strategyproof and then find its approximation guarantees.

Theorem 2. *The greedy mechanism for single-minded auctions is strategyproof.*

Proof. Firstly, note that it is never beneficial for any player i to misreport a bundle T_i' that does not contain T_i, because they will receive a value of 0 no matter what others report. Thus, it must be the case that $T_i \subseteq T_i'$.

Fixing a (potentially misreported) bundle $T_i' \supseteq T_i$, we now consider possible misreports of the valuation, w_i'. Player i's allocation is monotone weakly increasing in w_i'. In particular, by Lemma 1, for $w_i' < w_i^c(T_i)'$ player i does not get allocated any items while for $w_i' \geq w_i^c(T_i')$, player i gets allocated exactly all of T_i'.

Similarly to the proof of Vickrey auctions being strategyproof, as seen in the last lecture, we can now argue that it is never beneficial for player i to report a valuation $w_i' \neq w_i$. Note that player i changing their reported valuation w_i' only affects the allocation rule, but does not affect their payment if they are allocated their reported target bundle. Thus, it suffices to consider the cases where misreporting the valuation leads to a different allocation. First, consider a misreport of $w_i' > w_i$. Player i can only go from not being allocated anything to being allocated T_i'. In that case we know that $w_i' \geq w_i^c(T_i') \geq w_i$, so player i pay at least as much as their true valuation. Thus, their utility is at most 0, no better than if they weren't allocated anything. Next, consider a misreport of $w_i' < w_i$. Player i can only go from being allocated T_i' to not being allocated anything. In that case we know that $w_i \geq w_i^c(T_i') \geq w_i'$, the utility of player i goes from at least 0 (receiving a bundle they value at w_i for a payment of $w_i^c(T_i')$) to 0 (not being allocated anything). In both cases, player i can not benefit from misreporting their valuation w_i.

Finally, we show that misreporting a larger bundle $T_i' \supset T_i$ is never beneficial. Player i receives the same value from T_i and T_i', while the conflict set gets no smaller, $N_i'(T_i') \supseteq N_i'(T_i)$. Therefore, the critical value does not decrease, $w_i^c(T_i') \geq w_i^c(T_i)$, from the misreport. Thus, when misreporting a target bundle $T_i' \supset T_i$, they are either no longer allocated anything (going from a non-negative utility to zero utility) or if they still are allocated, their payment did not decrease.

In conclusion, a player misreporting their valuation and/or target bundle is never beneficial. □

Let's next formalize the framework for approximation algorithms.

Definition 5 (Approximation Algorithms)**.** An algorithm is a *c-approximation algorithm* (with $c \leq 1$) for an objective function in a maximization problem (e.g., social welfare) if for all instances $\mathcal{I}$ it holds that

$$\text{ALG}(\mathcal{I}) \geq c \cdot \text{OPT}(\mathcal{I}),$$

where $\text{ALG}(\mathcal{I})$ is the objective function value of the solution produced by the algorithm, and $\text{OPT}(\mathcal{I})$ is the optimal (maximal) objective function value of any solution.

An algorithm is a *c-approximation algorithm* (with $c \geq 1$) for an objective function in a minimization problem (e.g., social cost) if for all instances $\mathcal{I}$ it holds that

$$ALG(I) \leq c \cdot OPT(I).$$

In other words, the worst-case utility of the solution found by the algorithm will be within a multiplication factor of c of the optimum. In the case of a maximization problem, we are guaranteed at least a c-fraction of the optimum; in the case of minimization we are guaranteed that our solution is at most c times the optimum.[1]

Theorem 3. *The greedy mechanism for single-minded auctions is a $1/d$-approximation algorithm to social welfare, where d is the maximum size of any target bundle.*

[1] Beware that some sources may refer to what we call a c-approximation algorithm here as a $1/c$-approximation algorithm. For example, in Theorem 3, some authors would refer to the greedy mechanism as a d-approximation (instead of $1/d$). However, since we know that the approximation constant c is at most 1 for a maximization problem and at least 1 for a minimization problem — we cannot do better than the optimum — it should always be clear from the context what approximation guarantee the authors refer to.

Proof. Let N_{alg} be the set of winners by the greedy mechanism and let N_{opt} be the set of winners in the optimal solution. For any player $i \in N_{\text{alg}} \setminus N_{\text{opt}}$ that is only a winner under the greedy mechanism, let N_i be the set of players $j \in N_{\text{opt}}$ that they blocked from being allocated, i.e., where $w_j \leq w_i$ and $T_i \cap T_j \neq \emptyset$. For any player $i \in N_{\text{opt}} \cap N_{\text{alg}}$ that is a winner both in the greedy and the optimal solution, let $N_i = \{i\}$.

Now consider an arbitrary N_i. We know that

$$\sum_{j \in N_i} w_j \leq \sum_{j \in N_i} w_i \leq |N_i| w_i \tag{1}$$

since either $N_i = \{i\}$ or N_i contains players j with $w_j \leq w_i$. Furthermore, we know that the T_j's of the players $j \in N_i$ are disjoint from each other, since all $j \in N_i$ are part of the optimal solution N_{opt} with no overlaps in the allocation. However, we also know that for each player $j \in N_i$, the intersection $T_i \cap T_j$ contains at least one item. Thus, each T_j needs to overlap with a distinct element of T_i, so we get that

$$|N_i| \leq |T_i| \leq d. \tag{2}$$

In addition, note that

$$N_{\text{opt}} = \bigcup_{i \in N_{\text{alg}}} N_i. \tag{3}$$

Let's briefly prove both directions. For $N_{\text{opt}} \subseteq \bigcup_{i \in N_{\text{alg}}} N_i$, consider any player $j \in N_{\text{opt}}$. If they were also a winner in the greedy mechanism, they are in $N_j = \{j\}$. Else, there was some winner $i \in N_{\text{alg}}$ such that $w_j \leq w_i$ and $T_i \cap T_j \neq \emptyset$ that prevented the allocation to player j. In that case, we know that $j \in N_i$. The other direction $N_{\text{opt}} \supseteq \bigcup_{i \in N_{\text{alg}}} N_i$ follows immediately from the definition of each N_i as a subset of N_{opt}.

Putting it all together, with OPT as the welfare of the optimal solution and ALG as the welfare of the greedy mechanism, we get that

$$\begin{aligned}
\text{OPT} &= \sum_{i \in N_{\text{opt}}} w_i && \text{by definition} \\
&\leq \sum_{i \in N_{\text{alg}}} \sum_{j \in N_i} w_j && \text{from Equation (3)} \\
&\leq \sum_{i \in N_{\text{alg}}} d \cdot w_i && \text{from Equations (1) and (2)} \\
&= d \cdot \sum_{i \in N_{\text{alg}}} w_i \\
&= d \cdot \text{ALG} && \text{by definition.}
\end{aligned}$$

For the first inequality, note that we may count some elements of N_{opt} multiple times, which is why it is an inequality instead of an equality. □

We can also see that the greedy mechanism is no better than a $1/d$-approximation to social welfare.

Example 2 (Lower bound on greedy mechanism approximation)**.** Assume there are $d \leq n - 1$ items. Let the first d players desire exactly one distinct item each, $T_i = \{i\}$, with valuation $w_i = 1$, for all $i = 1, ..., d$, and let the $(d + 1)$th player desire all items, $T_{d+1} = \{1, ..., d\}$, with valuation $w_{d+1} = 1 + \varepsilon$ for some small $\varepsilon > 0$. All remaining players have empty target bundles and no valuation, $T_i = \emptyset$ and $w_i = 0$ for $i > d + 1$.

The greedy mechanism will first consider player $d + 1$ and allocate all goods to them, for a social welfare of $1 + \varepsilon$. However, optimally each of the first d players would be allocated the single item in their bundle for a social welfare of d. Thus, we see that there are cases where the greedy mechanism gives no better than a $1/d$-approximation to social welfare.

We illustrate the construction for $d = 4$, $n = 5$ below.

	1	2	3	4	5
a	♥				♥
b		♥			♥
c			♥		♥
d				♥	♥
w_i	1	1	1	1	$1+\varepsilon$
Greedy	✗	✗	✗	✗	✓
Optimal	✓	✓	✓	✓	✗

The main issue of the greedy algorithm in the above example is that it does not at all consider the size of the target bundle of each player. Intuitively, if two players have similar valuations, it should lead to better solutions to allocate to the player with the smaller target bundle (taking away less items from the other players). In particular, a variant of the greedy mechanism that considers the players in order of decreasing $(w_i/\sqrt{|T_i|})$ instead of w_i gives a $1/\sqrt{m}$-approximation for m items, improving on the $1/m$-approximation (where m is the worst-case d) from the greedy mechanism we considered so far. This likely is as good of an approximation as we can hope for in polynomial time.

Theorem 4 (NP-Hardness of Better Approximation)**.** *Unless* $\boldsymbol{P = NP}$*, there exists no polynomial-time computable, strategyproof mechanism that gives a c-approximation to social welfare with* $c > 1/\sqrt{m}$*.*

3 Task Assignment

We now consider another setting in which finding the welfare-maximizing solution is **NP**-hard.

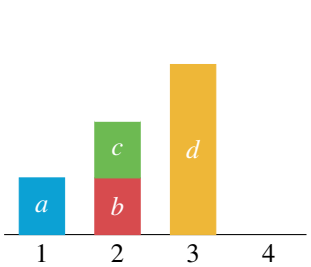
Definition 6 (Task Assignment)**.** An instance of a *task assignment* problem consists of

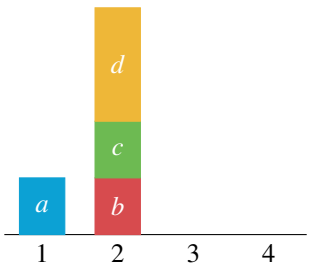
- a set G of m *tasks*,
- a set of n players N, and
- for each player $i \in N$ and task $j \in G$, the time c_{ij} that it takes player i to complete task j.

An allocation $\boldsymbol{A} = (A_1, ..., A_n)$, each task $j \in G$ is in exactly one A_i, allocates the tasks in A_i to player i. The cost of player i is the total time they need to spend, $\sum_{j \in A_j} c_{ij}$. Their utility is $-p_i - \sum_{j \in A_j} c_i j$ where p_i is the payment they receive. The *makespan* of an assignment is the longest time it will take any player to finish; that is, $\max_{i \in N} \sum_{j \in A_j} c_{ij}$. The social welfare is the negative of the sum of all the costs, $-\sum_{i \in N} \sum_{j \in A_i} c_{ij}$.

Example 3 (Task Assignment)**.** Consider a task assignment problem with tasks $G = \{a, b, c, d\}$, players $\{1, 2, 3, 4\}$, and the times for each player and task given by the table below. The plot in the middle shows the allocation $(\{a\}, \{b, c\}, \{d\}, \{\})$ with minimal makespan 3 and social welfare -6. The plot on the right shows the allocation $(\{a\}, \{b, c, d\}, \{\}, \{\})$ with maximal social welfare -5 and makespan 4.

	1	2	3	4
a	1	10	10	10
b	10	1	10	10
c	10	1	10	10
d	10	2	3	10





For many applications, we care about minimizing the makespan instead (instead of maximizing social welfare). However, we can show that the corresponding decision problem is **NP**-hard, implying that unless **P** = **NP**, no polynomial-time algorithm for finding a minimum makespan allocation exists.

Theorem 5. *Given an instance of task assignment and an integer k, deciding whether there exists an allocation with makespan at most k is **NP**-complete.*

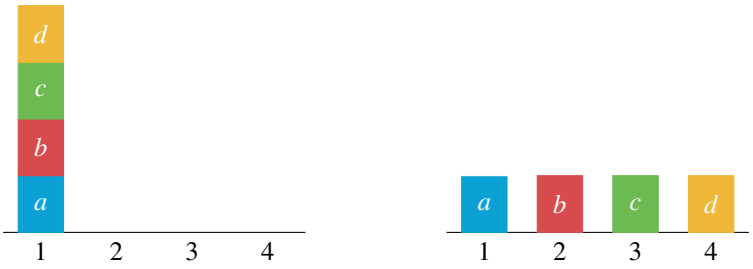
We defer a very brief but optional proof sketch to the Appendix.

While minimizing the makespan is intractable, we can efficiently compute an allocation maximizing social welfare by allocating each task to a player that takes the least time for it, i.e., a player i with $i \in \arg\min_{i \in N} c_{ij}$. Thus, we can implement the VCG mechanism for task assignment in polynomial time. However, the VCG mechanism is only an n-approximation algorithm for minimizing makespan.

Example 4 (Lower bound on VCG approximation to makespan)**.** Assume that there are n players and n tasks. Player 1 needs time $1 - \varepsilon$ for any task, so $c_{1j} = 1 - \varepsilon$ for all $j \in G$, where $\varepsilon > 0$ is a small constant. All other players need time 1 for all tasks, so $c_{ij} = 1$ for all $i \in N \backslash \{1\}$ and $j \in G$.

The solution with maximum social welfare is to assign all tasks to player 1, since they need the shortest time for each task. This gives a makespan of $n(1 - \varepsilon)$. However, if we assign each player one task, we can reach a makespan of 1. Thus, the VCG mechanism is no better than an n-approximation to the problem of minimizing the makespan.

We illustrate the construction for $n = 4$ below, with the allocation maximizing social welfare on the left and the allocation minimizing makespan on the right.



In 1999, Noam Nisan and Amir Ronen conjectured that this approximation ratio of n for the makespan is optimal for strategyproof mechanisms (even those running in exponential time). This was confirmed by George Christodoulou, Elias Koutsoupias, and Annamária Kovács in 2023.

Appendix

Theorem 5. *Given an instance of task assignment and an integer k, deciding whether there exists an allocation with makespan at most k is **NP**-complete.*

One can prove that this is the case already for only 2 players with a reduction from the Partition problem.

Definition 7 (Partition)**.** Given a set $S = \{x_1, ..., x_m\}$ of m positive integers that sum to T, is it possible to partition S into two subsets S_1 and S_2 so that $\sum_{i \in S_1} x_i = \sum_{i \in S_2} x_i = T/2$?

For the reduction, we can create an instance of task assignment with two players and m tasks, where $c_{1j} = c_{2j} = x_j$. There exists an allocation with makespan $T/2$ if and only if the tasks can be split up between the two players so that both have cost exactly $T/2$. Such an allocation corresponds to a partition of S into two subsets of sum $T/2$.