CSC2420 Spring 2017: Algorithm Design, Analysis and Theory
Lecture 12

Allan Borodin

April 3, 2017

Announcements

Announcements

1. There is an interesting talk this Friday at 10:30 at the Fields Institute (this years Avner Magen memorial lecture). You might want to meet the speaker whose research interests include spectral algorithms, spectral graph theory, convex programming, and approximation algorithms. Here is the title for her talk:
   Title: Strongly refuting random constraint satisfaction problems below the spectral threshold.
   The link for the talk is
   http://www.fields.utoronto.ca/activities/16-17/Magen-lecture
   Light refershments following the talk. After the talk or in the afternoon, you might want to meet the speaker whose research interests include spectral algorithms, spectral graph theory, convex programming, and approximation algorithms.
2. I have posted the first two questions for Assignment 3. I am planning to post more questions in a day or two and make assignment due in about 2 weeks.

Todays agenda

Todays agenda

1. A mention of some of the things we didn't mention (or barely mentioned) in the course.
    1. Maximizing a set function subject to a cardinality constraint
    2. We will mention some of that activity in game theory/mecahnism design, social choice theory and (even) social networks.
    3. Maximizing modula (i.e. linear) and monotone submodular function subject to matroid (and other independence) constraints.
2. (If time permits) Some things that I think could be research topics.

Set function maximization

In our MapReduce discussion, we mentioned the densest subgraph problem and earlier in the course (Lecture 6) we discussed max-cut in the context of maximizing a non-monotone submodular function. In Lecture 6 we also briefly discussed the SDP based algorithm (i.e. vector programs) for Max-2-Sat and mentioned that the same idea applied to max-cut.

Densest subgraph, max cut, max-di-cut and max-sat are problems where the goal is to maximize a non-monotone set function and hence (given that they are non-monotone) make sense in their *unconstrained* version.

Of course, similar to monotone set function maximization problems (such as max coverage), these problems also have natural constrained versions, the most studied version being a cardinality constraint.

More generally, there are other specific and arbitrary matroid constraints, other independence constraints, and knapsack constraints.

Cardinality constrained set function maximization

Max-k-densest subgraph, max-k-cut, max-k-di-cut, max-k-uncut and max-k-vertex-coverage are the natural cardinality constrained versions of well studied graph maximization problems. They all are of the following form:

Given an edge weighted graph $G = (V, E, w)$ with non negative edge weights $w : E \to \mathbb{R}$, find a subset $S \subseteq V$ with $|S| = k$ so as to maximize some set function $f(S)$. (Of course, In the unweighted versions, $w(e) = 1$ for all $e \in E$.)

For example, the objective in max-k-uncut is to find S so as to maximize the edge weights of the subgraphs induced by S and $V \setminus S$. That is, in a social network, divide the graph into two "communities".

NOTE: Max-k-sat is *not* the cardinality constrained version of max-sat in the same sense as the above problems. Although not studied (as far as I know), the analogous problem would be to find a set of propositional variables of cardinality k so as to maximize the weights of the satisfied clauses.

The SDP/vector program algorithms for the cardinality constrained problems

In what follows, I will briefly sketch some of the SDP based analysis in Feige and Langberg [2001]. This paper was proceeded and followed by a substantial number of important papers including the seminal Goemans and Williamson [1995] SDP approximation algorithm for max-cut.

(See , for example, Feige and Goemans [1995] and Frieze and Jerrun [1997] for proceeding work and Halperin and Zwick [2002], Han et al [2002] and Jäger and Srivastav for some improved and unifying results.)

There are also important LP based results such as the work by Ageev and Sviridenko [1999, 2004] that introduced *pipage rounding*.
Many papers focus on the *bisection* versions where $k = n/2$ and also $k = \sigma n$ for some $0 < \sigma < 1$ for which much better approximations atre known relative to results for a general cardinality k constraint.

The Goemans and Williamson program algorithm for max-cut

As stated in Lecture 6, vector programs can be solved to arbitrary precision within polynomial time.

- We introduce {-1,1} variables y_i corresponding to the vertex variables x_i. We also need a homogenizing variable y_0; the intended meaning is that vertex $v_i \in S$ and iff $y_i = y_0$.
- The max-cut problem can then be represented by the following (strict) quadratic programming problem:

  Maximize $\frac{1}{2}\sum_{1\leq i<j\leq n} w_{ij}(1 - y_i y_j)$ subject to
  $y_i^2 = 1$ for $0 \leq i \leq n$
- This is relaxed to a vector program by introducing vectors on the unit sphere in $\mathbf{v}_i \in \mathbb{R}^{n+1}$ where now the scalar multiplication becomes the vector inner product.

The rounding of the vector program

- The randomized rounding (from $\mathbf{v}_i^*$ to y_i) proceeds by choosing a random hyperplane in $\Re^{n+1}$ and then setting $y_i = 1$ iff $\mathbf{v}_i^*$ is on the same side of the hyperplane as $\mathbf{v}_0^*$. That is, if $\mathbf{r}$ is a uniformly random vector in $\Re^{n+1}$, then set $y_i = 1$ iff $\mathbf{r} \cdot \mathbf{v}_i^* \geq 0$.
- The rounded solution then has expected value $\sum_{1 \leq i < j \leq n} w_{ij} \mathbf{Pr}[\mathbf{v_i} \text{ and } \mathbf{v_j} \text{ are separated}] = \sum_{1 \leq i < j \leq n} w_{ij} \frac{\theta_{ij}}{\pi}$ where θ_{ij} is the angle between $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$.

The approximation ratio (in expectation) of the rounded solution

Let $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta)} \approx .87856$ and let OPT_{VP} be the value obtained by an optimal vector program solution.
Then $\mathbf{E}$[rounded solution] $\geq \alpha \cdot (OPT_{VP})$.

Extending the vector program formulation for the cardinality constraint

The basic idea is to add an additional constraint:

$$\sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_0 = 2k - n$$

It turns out that it is sometimes important to define an improved relaxation by using instead (for all $j \in \{0, \ldots, n\}$) the "caridnality constraints": $\sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_j = \mathbf{v_j} \mathbf{v}_0 (2k - n)$
For vectors $\mathbf{v}_j$ in the unit sphere, these constraints are equivalent to the constraints $\sum_{i=1}^{n} \mathbf{v}_i = \mathbf{v}_0 (2k - n)$

It also turns out that sometimes problems also use the following "triangle inequality constraints": $\mathbf{v}_i \mathbf{v}_j + \mathbf{v}_j \mathbf{v}_k + \mathbf{v}_k \mathbf{v}_i \geq -1$
and $\mathbf{v}_i \mathbf{v}_j - \mathbf{v}_j \mathbf{v}_k - \mathbf{v}_k \mathbf{v}_i \geq -1$ for all $i, j, k \in \{0, 1, \ldots, n\}$

What else has to be done?

Skipping some important considerations (not used for the max-k-coverage and max-k-densest subgraph problems) regarding how to merge this SDP/vector program relaxation with the LP max-cut formulation by Ageev and Svridenko, there is one very essential consideration that we have ignored thus far.

What else has to be done?

Skipping some important considerations (not used for the max-k-coverage and max-k-densest subgraph problems) regarding how to merge this SDP/vector program relaxation with the LP max-cut formulation by Ageev and Svridenko, there is one very essential consideration that we have ignored thus far.

The random hyperplane rounding insures the required probability that the round vectors will be separated. **BUT** this rounding does *not* enforce the desired k cardinality constraint.

What else has to be done?

Skipping some important considerations (not used for the max-k-coverage and max-k-densest subgraph problems) regarding how to merge this SDP/vector program relaxation with the LP max-cut formulation by Ageev and Svridenko, there is one very essential consideration that we have ignored thus far.

The random hyperplane rounding insures the required probability that the round vectors will be separated. **BUT** this rounding does *not* enforce the desired k cardinality constraint.

This is rectified by Feige and Langberg by modifying the SDP results so as to penalize the resulting sets S by a penalty depending on the deviation from the desired cardinality k.

What else has to be done?

Skipping some important considerations (not used for the max-k-coverage and max-k-densest subgraph problems) regarding how to merge this SDP/vector program relaxation with the LP max-cut formulation by Ageev and Svridenko, there is one very essential consideration that we have ignored thus far.

The random hyperplane rounding insures the required probability that the round vectors will be separated. **BUT** this rounding does *not* enforce the desired k cardinality constraint.

This is rectified by Feige and Langberg by modifying the SDP results so as to penalize the resulting sets S by a penalty depending on the deviation from the desired cardinality k.

Namely, they run the SDP sufficiently many times and output the set that maximizes $Z = \frac{w(S)}{OPT_{SDP}} + \theta_1 \frac{n-|S|}{n-k} + \theta_2 \frac{|S|(2k-|S|)}{n^2}$ where $w(S)$ is the SDP rounded output, OPT_{SDP} is the optimum relaxed value, and the θ are appropriately optimized scalars.

The formulation for other set function maximization problems

The formulation and idea of the relaxation follows the same idea by mainly changing the objective function. (Recall the objective fax max-2-sat.)

- For the max-k-densest subgraph problem, the objective (wrt. $y_i \in \{-1, 1\}$) is to maximize $\sum_{e_{ij} \in E} w_{ij} \frac{1+y_i y_0 + y_j y_0 + y_i y_j}{4}$
- The max-k-vertex-coverage problem a special case of the max coverage where each element (i.e. an edge) occurs in exactly two of the sets (i.e. vertices).

  The objective is to maximize $\sum_{e_{ij} \in E} w_{ij} \frac{3+y_i y_0 + y_j y_0 - y_i y_j}{4}$
  Here by monotocity we do not have to worry about outputs with $|S| < k$

  Now to compensate for $|S| > k$, we optimize $Z = \frac{w(S)}{OPT_{SDP}} + \theta \frac{n-|S|}{n-k}$.

The results in Feige and Langberg

TABLE 1
Approximation ratios achieved on our four maximization problems. Our results appear in columns in which the SDP technique is mentioned.

Problem	Technique	Approximation ratio	Range
Max-VC$_k$	Random	$1-(1-k/n)^2$	all k
	Greedy	$\max(1-1/e, 1-(1-k/n)^2)$	all k
	LP	$\frac{3}{4}$	all k
	SDP	0.8	$k \geq n/2$
	SDP	0.8	k size of minimum VC
	SDP	$3/4+\varepsilon$	all k, universal $\varepsilon > 0$
Max-DS$_k$	Random	$\frac{k(k-1)}{n(n-1)}$	all k
	Greedy	$O(k/n)$	all k
	LP	$\frac{k}{n}(1-\varepsilon)$	all k, every $\varepsilon > 0$
	SDP	$k/n+\varepsilon_k$	$k \sim n/2$
Max-Cut$_k$	Random	$\frac{2k(n-k)}{n(n-1)}$	all k
	LP	$\frac{1}{2}$	all k
	SDP	$1/2+\varepsilon$	all k, universal $\varepsilon > 0$
Max-UC$_k$	Random/LP	$1-\frac{2k(n-k)}{n(n-1)}$	all k
	SDP	$1/2+\varepsilon_k$	$k \sim n/2$

The results in Jäger and Srivastav

I think the following represents the latest improvements in cardinality constrained set function maximization for $k = \sigma n$ from Jäger and Srivastav [2005]

Problem	σ	Prev.	Our Method
MAX-k-CUT	0.3	0.527	0.567
MAX-k-UNCUT	0.4	0.5258	0.5973
MAX-k-DIRECTED-CUT	0.5	0.644	0.6507
MAX-k-DIRECTED-UNCUT	0.5	0.811	0.8164
MAX-k-DENSE-SUBGRAPH	0.2	0.2008	0.2664
MAX-k-VERTEX-COVER	0.6	0.8453	0.8784

Table 1: Examples for the improved approximation factors

Algorithm design everywhere

It is fair to say that almost every discipline has now been significantly impacted by computation (and, in particular, by algorithms of one form or another.) But for me, the question is to what extent does CS bring a unique perspective to another discipline? (And, of course, one should consider how CS has been significantly influenced by other disciplines.)

Algorithm design everywhere

It is fair to say that almost every discipline has now been significantly impacted by computation (and, in particular, by algorithms of one form or another.) But for me, the question is to what extent does CS bring a unique perspective to another discipline? (And, of course, one should consider how CS has been significantly influenced by other disciplines.)

Computing had its start in data processing (becoming data base and information systems) and scientific computing (mainly in the physical and mathematical sciences). And in turn these fields provided much insight and results adapted within CS. More recently (relative to the initial applications) has been the impact of CS in the biological sciences.

Algorithm design everywhere

It is fair to say that almost every discipline has now been significantly impacted by computation (and, in particular, by algorithms of one form or another.) But for me, the question is to what extent does CS bring a unique perspective to another discipline? (And, of course, one should consider how CS has been significantly influenced by other disciplines.)

Computing had its start in data processing (becoming data base and information systems) and scientific computing (mainly in the physical and mathematical sciences). And in turn these fields provided much insight and results adapted within CS. More recently (relative to the initial applications) has been the impact of CS in the biological sciences.

And perhaps even more recently (ie. last 15 years), the impact has also spread to the social sciences. What, if any, is the unique perspective that CS brings to (say) a field such as game theory/mechanism design which is a well developed subject grounded in economics and mathematics?

What is mechanism design and social choice theory?

In game theory, we are given a "game" and the goal is to understand and analyze the strategies of agents and deduce the resulting outcome(s).

What is mechanism design and social choice theory?

In game theory, we are given a "game" and the goal is to understand and analyze the strategies of agents and deduce the resulting outcome(s).

The raison d'etre of mechanism design is to design mechanisms (for example, algorithms to allocate and price sets of items) so as to induce games with desireable outcomes. It is therefore sometimes called *inverse game theory*.

What is mechanism design and social choice theory?

In game theory, we are given a "game" and the goal is to understand and analyze the strategies of agents and deduce the resulting outcome(s).

The raison d'etre of mechanism design is to design mechanisms (for example, algorithms to allocate and price sets of items) so as to induce games with desireable outcomes. It is therefore sometimes called *inverse game theory*.

The agents in game theory and mechanism design traditionally have cardinal (e.g. monetary values) utilities wheres in **social choice theory** (e.g. voting, peer evaluation), agents usually have preferences. With that possible distinction in mind, we can view social choice theory as part of game theory/mechanism design.

More on the nature of mechanism design

In mechanism design, one designs games where preferences (or payoffs) are unknown (i.e. private information), so when players act rationally "desirable outcomes" emerge. (Note: It is often argued as to if and when individuals act rationally.)

More on the nature of mechanism design

In mechanism design, one designs games where preferences (or payoffs) are unknown (i.e. private information), so when players act rationally "desirable outcomes" emerge. (Note: It is often argued as to if and when individuals act rationally.)

Mechanism design is a type of algorithm design where private inputs are coming from self interested strategic agents.

More on the nature of mechanism design

In mechanism design, one designs games where preferences (or payoffs) are unknown (i.e. private information), so when players act rationally "desirable outcomes" emerge. (Note: It is often argued as to if and when individuals act rationally.)

Mechanism design is a type of algorithm design where private inputs are coming from self interested strategic agents.

Traditionally, mechanism design involves money (to some extent) to incentivize agents to choose strategies leading to a desireable outcome.

More on the nature of mechanism design

In mechanism design, one designs games where preferences (or payoffs) are unknown (i.e. private information), so when players act rationally "desirable outcomes" emerge. (Note: It is often argued as to if and when individuals act rationally.)

Mechanism design is a type of algorithm design where private inputs are coming from self interested strategic agents.

Traditionally, mechanism design involves money (to some extent) to incentivize agents to choose strategies leading to a desireable outcome.

There is also an area of *mechanism design without money*. This includes stable matching, fair division, and voting theory. See, for example, see Chapters 11,12,13 in the game theory text by Karlin and Peres.

Auctions

One prominent part of mechanism design concerns auctions. A reasonably general auction setting consists of the following ingrediants:

- A set (or multiset) M of items to be sold.
- A set U of buyers having valuations for various sets (or multisets) of items and possibly a budget.
- A number of sellers having costs for producing items.
- The outcome of an auction mechanism is a "feasible allocation" of the items to the buyers so as to achieve certain desired goals. That is, there are contraints on what allocations are feasible.

The above formulation precludes *externalities.*)

Auctions and mechanism design

If all information is publicly known, then this is a pure combinatorial problem that requires an algorithm to compute an optimal or near optimal desireable allocation. There are no strategic considerations.

Auctions and mechanism design

If all information is publicly known, then this is a pure combinatorial problem that requires an algorithm to compute an optimal or near optimal desireable allocation. There are no strategic considerations.

But in mechanism design we assume that some information (e.g. the valuations and/or budgets of the buyers, the costs of the sellers) is private.

Auctions and mechanism design

If all information is publicly known, then this is a pure combinatorial problem that requires an algorithm to compute an optimal or near optimal desireable allocation. There are no strategic considerations.

But in mechanism design we assume that some information (e.g. the valuations and/or budgets of the buyers, the costs of the sellers) is private.

Because agents (e.g. buyers or sellers) are strategic they may not be truthful in reporting their private information. A mechanism needs to incentivize strategic behaviour to achieve a desireable outcome given only partial or no prior knowledge of the private information.

Auctions and mechanism design

If all information is publicly known, then this is a pure combinatorial problem that requires an algorithm to compute an optimal or near optimal desireable allocation. There are no strategic considerations.

But in mechanism design we assume that some information (e.g. the valuations and/or budgets of the buyers, the costs of the sellers) is private.

Because agents (e.g. buyers or sellers) are strategic they may not be truthful in reporting their private information. A mechanism needs to incentivize strategic behaviour to achieve a desireable outcome given only partial or no prior knowledge of the private information.

In Bayesian mechanism design, the mechanism (and perhaps the agents) have prior common distributional knowledge about the private information.

Auctions and mechanism design

If all information is publicly known, then this is a pure combinatorial problem that requires an algorithm to compute an optimal or near optimal desireable allocation. There are no strategic considerations.

But in mechanism design we assume that some information (e.g. the valuations and/or budgets of the buyers, the costs of the sellers) is private.

Because agents (e.g. buyers or sellers) are strategic they may not be truthful in reporting their private information. A mechanism needs to incentivize strategic behaviour to achieve a desireable outcome given only partial or no prior knowledge of the private information.

In Bayesian mechanism design, the mechanism (and perhaps the agents) have prior common distributional knowledge about the private information.

The objective of a mechanism may be to try to optimize social welfare (also called social surplus) or the mechanism may itself be viewed as an agent trying to maximize its revenue.

So what, if any, is the unique perspective that CS brings to mechanism design?

Nisan and Ronen [1999] established Algorithmic game theory (AGT) by the recognition that the kinds of large scale applications that are now possible (i.e. online auctions) requires computationally efficient algorithms. Given that we strongly believe that optimality is not efficiently obtainable for many combinatorial problems, we have to settle for approximations.

So what, if any, is the unique perspective that CS brings to mechanism design?

Nisan and Ronen [1999] established Algorithmic game theory (AGT) by the recognition that the kinds of large scale applications that are now possible (i.e. online auctions) requires computationally efficient algorithms. Given that we strongly believe that optimality is not efficiently obtainable for many combinatorial problems, we have to settle for approximations.

It is this computational awareness and the interest in approxmation that was a rather unusual (if not unique) perspective coming from CS.

So what, if any, is the unique perspective that CS brings to mechanism design?

Nisan and Ronen [1999] established Algorithmic game theory (AGT) by the recognition that the kinds of large scale applications that are now possible (i.e. online auctions) requires computationally efficient algorithms. Given that we strongly believe that optimality is not efficiently obtainable for many combinatorial problems, we have to settle for approximations.

It is this computational awareness and the interest in approxmation that was a rather unusual (if not unique) perspective coming from CS.

And now there is also an acknowledgement that agents (e.g. say the buyers) have to understand (and be able to execute) their possible strategies and hence, in practice, one often wants (or requires) conceptually simple auctions.

Combinatorial auctions: an example of a mechanism design problem

As we mentioned, the suggested framework is rather general and one usually considers more restrictive settings. Lets consider one setting that is perhaps the most studied in theoretical computer science.

The (direct revelation) combinatorial auction (CA) problem

In the CA problem, there is a set M of m items, a set U of n buyers, and one seller which we can also view as the Mechanism. Each agent i has a private valuation function $v_i : 2^M \to \mathbb{R}^{\geq 0}$. Each agent will submit bids $b_i(S) \geq 0$ for the subsets S it desires. The mechanism will allocate a desired subset S_i (possibly the empty subset) to each agent and will charge a price $p_i(S_i) \leq b_i(S_i)$ for the set allocated. The *quasi linear utility* of agent i for this allocation is $u_i(S_i) = v_i(S_i) - p_i(S_i)$. A feasible allocation is a collection of disjoint subsets S_i.

The CA problem continued

What is the goal or utility of the seller/mechanism?

The CA problem continued

What is the goal or utility of the seller/mechanism?

One possible goal is social welfare/surplus; namely, to obtain a feasible allocation that maximizes the values of the allocated sets.
Note: This is equivalent to maximizing the utilties of the allocated sets + the revenue collected by the seller/mechanism.

The CA problem continued

What is the goal or utility of the seller/mechanism?

One possible goal is social welfare/surplus; namely, to obtain a feasible allocation that maximizes the values of the allocated sets.
Note: This is equivalent to maximizing the utilties of the allocated sets + the revenue collected by the seller/mechanism.

A specific example of a CA is the wireless spectrum auction. Here we envision the government (i.e. the mechanism) is allocating licenses for various collections of spectrum frequencies and it is not completely unreasonable to assume that the goal of the mechanism is social welfare.

CA problem continued

In its generality, each agent has a value for each possible subset of items. This requires an exponential (in m) input representation. To algorithmically accomodate (possibly exponential) size set system problems, one can assume some sort of "oracle" such as a value oracle (as we discussed for the problem of maximizing a non-monotone submodular function.) that given S will return a value, say $b_i(S)$.

CA problem continued

In its generality, each agent has a value for each possible subset of items. This requires an exponential (in m) input representation. To algorithmically accomodate (possibly exponential) size set system problems, one can assume some sort of "oracle" such as a value oracle (as we discussed for the problem of maximizing a non-monotone submodular function.) that given S will return a value, say $b_i(S)$.

In practice, many CA problems can be represented explicitly and succinctly. For example, if each agent is interested in only one set. This is referred to as a *single-minded CA*. As long, as each agent is only interested in a few sets, the CA problem can be represented explicitly and succinctly.

CA problem continued

In its generality, each agent has a value for each possible subset of items. This requires an exponential (in m) input representation. To algorithmically accomodate (possibly exponential) size set system problems, one can assume some sort of "oracle" such as a value oracle (as we discussed for the problem of maximizing a non-monotone submodular function.) that given S will return a value, say $b_i(S)$.

In practice, many CA problems can be represented explicitly and succinctly. For example, if each agent is interested in only one set. This is referred to as a *single-minded CA*. As long, as each agent is only interested in a few sets, the CA problem can be represented explicitly and succinctly.

Another explictly represented CA is the sCA problem where every desired set has cardinality at most s (for some small constant s).

CA problem continued

In its generality, each agent has a value for each possible subset of items. This requires an exponential (in m) input representation. To algorithmically accomodate (possibly exponential) size set system problems, one can assume some sort of "oracle" such as a value oracle (as we discussed for the problem of maximizing a non-monotone submodular function.) that given S will return a value, say $b_i(S)$.

In practice, many CA problems can be represented explicitly and succinctly. For example, if each agent is interested in only one set. This is referred to as a *single-minded CA*. As long, as each agent is only interested in a few sets, the CA problem can be represented explicitly and succinctly.

Another explictly represented CA is the sCA problem where every desired set has cardinality at most s (for some small constant s).

Note: We assume that $v_i(\varnothing) = 0$ and that $v_i(S)$ is a monotone set function for every i (i.e. *free disposal*). Thus, sets that are not explicitly given values inherit their value from a desired subset of largest value.

The CA problem continued and what is *algorithmic* mechanism design

What is the algorithmic challenge of this problem?

The CA problem continued and what is *algorithmic* mechanism design

What is the algorithmic challenge of this problem?

The underlying allocation problem is the *set packing problem*; namely, given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$, where each set S_j has a value v_j, choose a subcollection $\mathcal{S}'$ of disjoint sets so as to maximize $\sum_{\{j:S_j \in \mathcal{S}'\}} v_j$.

The CA problem continued and what is *algorithmic* mechanism design

What is the algorithmic challenge of this problem?

The underlying allocation problem is the *set packing problem*; namely, given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$, where each set S_j has a value v_j, choose a subcollection $\mathcal{S}'$ of disjoint sets so as to maximize $\sum_{\{j:S_j \in \mathcal{S}'\}} v_j$.

As we previously discussed, this problem is NP-hard and NP-hard to approximatte to within a factor $m^{\frac{1}{2}-\epsilon}$, even when all $v_j = 1$ (i.e. the unweighted case).

The CA problem continued and what is *algorithmic* mechanism design

What is the algorithmic challenge of this problem?

The underlying allocation problem is the *set packing problem*; namely, given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$, where each set S_j has a value v_j, choose a subcollection $\mathcal{S}'$ of disjoint sets so as to maximize $\sum_{\{j:S_j \in \mathcal{S}'\}} v_j$.

As we previously discussed, this problem is NP-hard and NP-hard to approximatte to within a factor $m^{\frac{1}{2}-\epsilon}$, even when all $v_j = 1$ (i.e. the unweighted case).

Furthermore, the set packing problem is NP hard even when all sets have cartdinality at most s (i.e. the underlying allocation problem for the sCA problem) for $s \geq 3$ and hard to approximate to a factor $\Omega(\frac{s}{\ln s})$.

Strategic behaviour meets computational complexity

We will soon discuss *truthfulness* where bidding truthfully is a dominant strategy. A truthful mechanism (also referred to as an *incentive compatible IC or dominant strategy incentive compatible DSIC mechanism*) is one that results in truthful bidding being a dominant strategy. And now here is the issue that began algorithmic game theory. (See the seminal papers by Nisan and Ronen, and by Lehmann, O'Callahan and Shoham.)

If we could compute an optimal allocation (i.e. one optimizing social welfare), we could then rely upon Vickrey-Clarke-Groves (VCG) pricing (i.e. charge each agent the loss of social welfare caused by their presense in an optimal allocation) to obtain a truthful mechanism.

Strategic behaviour meets computational complexity

We will soon discuss *truthfulness* where bidding truthfully is a dominant strategy. A truthful mechanism (also referred to as an *incentive compatible IC or dominant strategy incentive compatible DSIC mechanism*) is one that results in truthful bidding being a dominant strategy. And now here is the issue that began algorithmic game theory. (See the seminal papers by Nisan and Ronen, and by Lehmann, O'Callahan and Shoham.)

If we could compute an optimal allocation (i.e. one optimizing social welfare), we could then rely upon Vickrey-Clarke-Groves (VCG) pricing (i.e. charge each agent the loss of social welfare caused by their presense in an optimal allocation) to obtain a truthful mechanism.

But the NP-hardness of this problem precludes an optimal allocation (at least in the worst case, if we assume P $\neq$ NP) and hence we would have to rely on some approximation algorithm.

Strategic behaviour meets computational complexity

We will soon discuss *truthfulness* where bidding truthfully is a dominant strategy. A truthful mechanism (also referred to as an *incentive compatible IC or dominant strategy incentive compatible DSIC mechanism*) is one that results in truthful bidding being a dominant strategy. And now here is the issue that began algorithmic game theory. (See the seminal papers by Nisan and Ronen, and by Lehmann, O'Callahan and Shoham.)

If we could compute an optimal allocation (i.e. one optimizing social welfare), we could then rely upon Vickrey-Clarke-Groves (VCG) pricing (i.e. charge each agent the loss of social welfare caused by their presense in an optimal allocation) to obtain a truthful mechanism.

But the NP-hardness of this problem precludes an optimal allocation (at least in the worst case, if we assume P $\neq$ NP) and hence we would have to rely on some approximation algorithm.

But VCG pricing does not always result in a truthful mechanism for approximation algorithms!

VCG mechanism = optimal allocation + VCG pricing.

The Vickrey auction for a single item

In a sealed auction for a single item, the auctioneer (the mechanism, the seller) receives bids $(b_1, \ldots, b_n)$ from (say) n bidders who have private values $(v_1, \ldots, v_n)$ for the item. Notably, the bids may not be equal to the values and the mechanism may not know anything about the true values (or it might know a prior distribution on these values).

The Vickrey auction for a single item

In a sealed auction for a single item, the auctioneer (the mechanism, the seller) receives bids $(b_1, \ldots, b_n)$ from (say) n bidders who have private values $(v_1, \ldots, v_n)$ for the item. Notably, the bids may not be equal to the values and the mechanism may not know anything about the true values (or it might know a prior distribution on these values).

Based on these bids, the mechanism allocates the item (i.e. determines the winner) and sets a price for the winner to pay. If the mechanism only wishes to maximize social welfare, what should it do?

The Vickrey auction for a single item

In a sealed auction for a single item, the auctioneer (the mechanism, the seller) receives bids $(b_1, \ldots, b_n)$ from (say) n bidders who have private values $(v_1, \ldots, v_n)$ for the item. Notably, the bids may not be equal to the values and the mechanism may not know anything about the true values (or it might know a prior distribution on these values).

Based on these bids, the mechanism allocates the item (i.e. determines the winner) and sets a price for the winner to pay. If the mechanism only wishes to maximize social welfare, what should it do?

For example, suppose I am interested in selling my legally unlocked 128GB iPhone 6. I am assuming you know what it is worth to you! I will announce my allocation and pricing algorithm and you will then bid. My goal is to make sure that the person who values it the most will be the winner of my auction. (I may also believe that this person will bid reasonably and I will get some revenue.) What should I do?

The Vickrey auction continued: selling my iPhone 6

1. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person their bid. This is the *first price auction*.

The Vickrey auction continued: selling my iPhone 6

1. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person their bid. This is the *first price auction*.

2. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person the second highest bid. This is the Vickrey second price auction.

The Vickrey auction continued: selling my iPhone 6

1. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person their bid. This is the *first price auction*.

2. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person the second highest bid. This is the Vickrey second price auction.

How many people bid their true value for the first mechanism? If not, what fraction of the true value did you bid?

The Vickrey auction continued: selling my iPhone 6

1. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person their bid. This is the *first price auction*.

2. I will allocate the phone to the person with the highest bid (ignore tying bids) and charge that person the second highest bid. This is the Vickrey second price auction.

How many people bid their true value for the first mechanism? If not, what fraction of the true value did you bid?

How many people bid their true value for the second mechanism? If not what fraction of the true value did you bid?

The second price auction is truthful

The Vickrey auction is truthful

Bidding truthfully is a dominant strategy (no matter how the other buyers bid) and reasonably assuming everyone bids truthfully, social welfare is maximized.

There is an obvious issue with both the first and second price auctions in that the auctioneer/seller may value the item more than any of the buyers.

The second price auction is truthful

The Vickrey auction is truthful

Bidding truthfully is a dominant strategy (no matter how the other buyers bid) and reasonably assuming everyone bids truthfully, social welfare is maximized.

There is an obvious issue with both the first and second price auctions in that the auctioneer/seller may value the item more than any of the buyers.

This is dealt with by the mechanism announcing a *reserve price* so that no bid will be accepted that is under the reserve price.

The English (ascending price) auction

While the second price auction may seem a little strange at first, it is in some sense equivalent to the familar English ascending price auction.

The English (ascending price) auction

While the second price auction may seem a little strange at first, it is in some sense equivalent to the familar English ascending price auction.

In the English auction, the auctioneer starts with a price (i.e. a reserve price) and then continues to ask who wants to raise the bid? The final (highest) bid wins.

The English (ascending price) auction

While the second price auction may seem a little strange at first, it is in some sense equivalent to the familar English ascending price auction.

In the English auction, the auctioneer starts with a price (i.e. a reserve price) and then continues to ask who wants to raise the bid? The final (highest) bid wins.

Suppose the auctioneer always just asks if there is anyone who wants to raise the current bid by some small ϵ. Then this ascending price auction is essentially producing the same outcome (assuming buyers do not change their valuation given other bids) in terms of who wins the item and what is paid. These are referred to as two different *implementations* of the same outcome.

The English (ascending price) auction

While the second price auction may seem a little strange at first, it is in some sense equivalent to the familar English ascending price auction.

In the English auction, the auctioneer starts with a price (i.e. a reserve price) and then continues to ask who wants to raise the bid? The final (highest) bid wins.

Suppose the auctioneer always just asks if there is anyone who wants to raise the current bid by some small ϵ. Then this ascending price auction is essentially producing the same outcome (assuming buyers do not change their valuation given other bids) in terms of who wins the item and what is paid. These are referred to as two different *implementations* of the same outcome.

But how do these two implementations differ?

Revenue equivalence

We have seen (if my iPhone 6 sale worked as it should have) that the buyers strategies are different for 1st price and 2nd price auctions. Which mechanism is best for me (the seller) in terms of my revenue?

Revenue equivalence

We have seen (if my iPhone 6 sale worked as it should have) that the buyers strategies are different for 1st price and 2nd price auctions. Which mechanism is best for me (the seller) in terms of my revenue?

Surpising fact: If we assume say that all bidders are drawing their values from same uniform distribution (say $U[0, h]$) then my expected revenue is the same!

Revenue equivalence

We have seen (if my iPhone 6 sale worked as it should have) that the buyers strategies are different for 1st price and 2nd price auctions. Which mechanism is best for me (the seller) in terms of my revenue?

Surpising fact: If we assume say that all bidders are drawing their values from same uniform distribution (say $U[0, h]$) then my expected revenue is the same!

Even more surprising: This revenue equivalence holds much more generally. There are some technical conditions involved but here is the informal statement in Wikipedia.

Revenue equivalence

We have seen (if my iPhone 6 sale worked as it should have) that the buyers strategies are different for 1st price and 2nd price auctions. Which mechanism is best for me (the seller) in terms of my revenue?

Surpising fact: If we assume say that all bidders are drawing their values from same uniform distribution (say $U[0, h]$) then my expected revenue is the same!

Even more surprising: This revenue equivalence holds much more generally. There are some technical conditions involved but here is the informal statement in Wikipedia.

Revenue equivalence is a concept in auction theory that states that given certain conditions, any mechanism that results in the same outcomes (i.e. allocates items to the same bidders) also has the same expected revenue.

Revenue equivalence

We have seen (if my iPhone 6 sale worked as it should have) that the buyers strategies are different for 1st price and 2nd price auctions. Which mechanism is best for me (the seller) in terms of my revenue?

Surpising fact: If we assume say that all bidders are drawing their values from same uniform distribution (say $U[0, h]$) then my expected revenue is the same!

Even more surprising: This revenue equivalence holds much more generally. There are some technical conditions involved but here is the informal statement in Wikipedia.

Revenue equivalence is a concept in auction theory that states that given certain conditions, any mechanism that results in the same outcomes (i.e. allocates items to the same bidders) also has the same expected revenue.

Note: I would replace "concept" by "result" (i.e. a theorem)

Cake cutting and fair division

Consider the problem of cutting (i.e. partitioning) a cake into some number n disjoint "pieces", say $A_1, \ldots, A_n$. We want to do this in a "fair way". By definition, we view our cake as being completely *divisible*. For our more mathematical/computational purpose, we can think of the cake as the unit interval $[0, 1]$

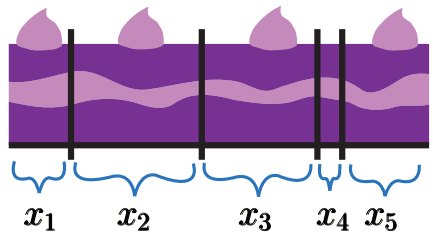

FIGURE 11.2. This figure shows a possible way to cut a cake into 5 pieces. The i^{th} piece is $B_i = [\sum_{k=1}^{i-1} x_k, \sum_{k=1}^{i} x_k)$. If the i^{th} piece goes to player j, then his value for this piece is $\mu_j(B_i)$.

Figure: Cutting a "cake" into 5 contiguous pieces

So what is a "piece" and what is "fair"? Maybe I only like certain parts of the cake while others like other parts? A "piece" is a finite collection of disjoint intervals. Pieces need not be contiguous.

Cake cutting definitions

We will assume that each agent (also called players) i has a valuation function $v_i : [0,1] \to \mathbb{R}^{\geq 0}$ that can be thought of as a cummulative distribution function. Namely,

1. For all $X \subseteq [0,1]$, $v_i(X) \geq 0$
2. For all $X, Y \subseteq [0,1] : X \cap Y = \varnothing$, $v_i(X \cup Y) = v_i(X) + v_i(Y)$
3. For all $x \in [0,1]$, $v_i(x,x) = 0$
4. $v_i([0,1]) = 1$

There are two natural definitions of "fair division":

- A division $A_1, \ldots, A_n$ is a *proportional division* if $v_i(A_i) \geq 1/n$ for all i. **Note:** This is called "fair" in the KP text but I prefer to use "fair" as a more intuitive concept.
- A division $A_1, \ldots, A_n$ is an *envy-free division* if $v_i(A_i) \geq v_i(A_j)$ for all i, j. iThat is, i does not envy j's allocation.

Proportional fairness vs envy-free fairness

The main concern is fairness. It turns out that envy-freeness is a more desireable property.
We need only consider $n \geq 2$ agents since $n = 1$ is trivial.

Every envy-free division is a proportional division

Proof: For any i, there must be some S_j such that $v_i(S_j) \geq 1/n$. Then since the division is envy-free, $v_i(A_i) \geq v_i(A_j) \geq 1/n$.

However, for $n \geq 3$, the converse is not necessarily true. Consider the following valuation profile:
$v_1([0, 1/3]) = 1/3, v_1([1/3, 2/3]) = 2/3, v_1([2/3, 1]) = 0$; v_2, v_3 have uniform valuations.
The division $A_1 = [0, 1/3), A_2 = [1/3, 2/3), A_3 = [2/3, 1]$ is proprtional since everyone is receiving a $1/3$ valuation but agent 1 envies the allocation to agent 2.

What other properties do we want in a fair cake cutting algorithm?

The following properties are all desireable but maybe not all are achieveable.

- The allocated pieces are contiguous.
- The protocol is truthful.
- The protocol is (approximately) socially optimal.
- The "complexity" of the cake-cutting algorithm is (approximately) optimal amongst envy-free or proportional divisions.

We need to define what measures of complexity we might want to consider. But before we do so, let us consider a natural algorithm for $n = 2$ agents. We will then see that the situation for $n \geq 3$ agent becomes more complicated.

The "Cut and Choose" algorithm for $n = 2$ agents

The following is a protocol (i.e. algorithm) you may have used and can be found in the Old Testement (see chapter notes):

The cut and choose algorithm

1. Agent 1 cuts the "cake" $[0, 1]$ into two equal parts according to his valuation; that is, $v_1(A_1) = v_1(A_2) = 1/2$.
2. Agent 2 chooses between A_1 and A_2.

What properties are satisfied by cut and choose?

The "Cut and Choose" algorithm for $n = 2$ agents

The following is a protocol (i.e. algorithm) you may have used and can be found in the Old Testement (see chapter notes):

The cut and choose algorithm

1. Agent 1 cuts the "cake" $[0, 1]$ into two equal parts according to his valuation; that is, $v_1(A_1) = v_1(A_2) = 1/2$.
2. Agent 2 chooses between A_1 and A_2.

What properties are satisfied by cut and choose?

- Envy-free?

The "Cut and Choose" algorithm for $n = 2$ agents

The following is a protocol (i.e. algorithm) you may have used and can be found in the Old Testement (see chapter notes):

The cut and choose algorithm

1. Agent 1 cuts the "cake" $[0, 1]$ into two equal parts according to his valuation; that is, $v_1(A_1) = v_1(A_2) = 1/2$.
2. Agent 2 chooses between A_1 and A_2.

What properties are satisfied by cut and choose?

- Envy-free? Yes. Agent 2 clearly gets the best of A_1 and A_2 and is hence envy-free; agent 1 chose the partition so as to have equal value so he is also envy-free. Division must then be proportional.
- Contiguous? Yes agent 1, can choose to "query" which location x satisfies $v_1[0, x) = v_1(x, 1]$
- Complexity? One cut which is clearly optimal. What else might we measure?
- Truthful? To be more precise "ex-post IC"?
- Socially optimal?

Complexity measures: a CS perspective

Lets look a little closer at the complexity of cut and choose. As we said, the number of cuts was optimal. More generally, for any number n of agents, the minimum possible number of cuts is $n-1$ and this is achieveable iff the division is contiguous.

In the cut and choose protocol, besides the number of cuts, there were two other queries that can be considered as possible meausres of complexity. Agent 1 asked where to cut so as to obtain a piece of value $1/2$ and agent 2 asked for his value of a piece. (Agent 2 only had to ask about one piece as that determined the value of both A_1 and A_2.).

Here is the Robertson and Web [1998] complexity model:

1. Agent 1 used a *demand query*; namely given some value v and some current piece X (i.e. X is an interval in $[0,1]$), the agent i asked where to cut the piece X so as to obtain a piece Y (i.e. Y is a subinterval of X) such that $v_i(Y) = v$.
2. Agent 2 used a *value query*; namely, given a piece X, the agent asked for his value on this piece.

The moving knife protocol: A proportional division for any number of agents

The moving knife protocol (due to Dubins and Spanier [1961]) is the following conceptually simple algorithm:

Moving knife protocol

Initialize: Let N be the set of n agents; $X := [0, 1]$;
start the knife at the leftmost location 0.
While $|N| > 1$
Move the knife to the right until some agent $i \in N$ yells "STOP" having observed value $v_i(Y) = 1/|N|$ for the piece Y to the left of the knife
Cut the "cake" and give agent i piece Y; $N := N \setminus \{i\}$; $X := X \setminus Y$
End While The one remaining player gets the remaining piece.

Equivalently, the KP text presents the algorithm recursively.

Properties of the moving knife protocol

The following properties are easy to verify

- The allocation is a contiguous division using the minimum $n-1$ cuts.
- The division is fair. Why?

Properties of the moving knife protocol

The following properties are easy to verify

- The allocation is a contiguous division using the minimum $n-1$ cuts.
- The division is fair. Why? Note that when the first cut is made, any one of the remaining $n-1$ agents (say agent j) has value for what remains $v_j(X \setminus Y) \geq 1 - \frac{1}{n} = \frac{n-1}{n}$ which has to be shared but now shared with only $n-1$ players. Hence (by induction) every agent gets a share with value at least $\frac{1}{n}$.
- More precisely, the first $n-1$ players to yell STOP get exactly a value of $\frac{1}{n}$ and the last player obtains a value at least $\frac{1}{n}$.
- The division is *not* necessarily envy-free for $n > 2$ agents as shown in figure 11.3 of the KP text (see next slide). What agent is guaranteed to not be envious?

The negative aspects of the moving knife protocol

As we just stated, the division may not be envy-free.

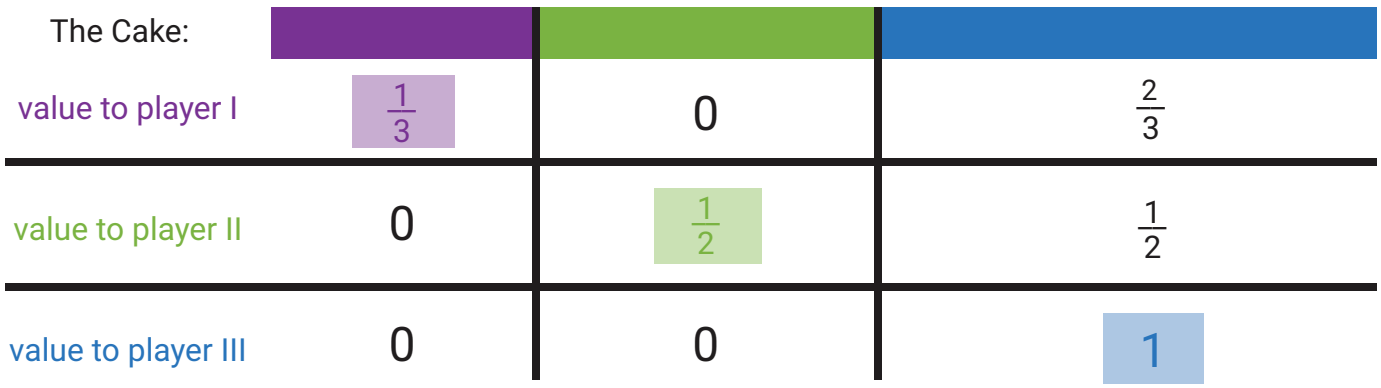

Figure: an envious division

In addition to not being envy-free, the moving knife requires an active referee who is slowly moving the knife. In other words this is not a discrete algorithm with a finite number of queries of the two types we have described. Each agent would need to be continuously asking for the value of the piece to the left of the knife. Evan and Paz [1984] adapted the moving knife so that it is a discrete algorithm in the Robertson and Web model using $O(n \log n)$ queries.

The continuing story for cake cutting for the $n > 2$ agents

All of the following non-trivial cake cutting protocols result in non-contiguous but discrete computation divisions. (Note: for $n > 2$, it is not possible in general to have an envy-free continuous divisions.)

- For $n = 3$, Steinhaus [1943] gave a protocol that ylelds a proportional division using at most 3 cuts. This protocol is not envy-free.
- For $n = 3$, Selfridge and Conway [1960] gave a envy-free protocol with at most 5 cuts.
- For $n = 4$, Brams and Taylor [1995] gave an envy-free protocol with an unbounded number of cuts. That is, for any instance this algorithm uses some finite number of cuts and queries but that number depends on the instance; that is, for all c, there is an instance requiring at least c cuts.
- Su [1999] gave a non constructive proof that there exist envy-free divisions for all n.

The latest developments in this continuing story of cake cutting

- After 20 more year, Aziz ans MacKenzie [2016, spring] gave an envy-free protocol for $n = 4$ agents with a bounded number of cuts.
- In the latest development, the same Aziz and McKenzie [2016, fall] gave an envy-free protocol for all n using a bounded number of cuts and queries. However, so far the bound on the number of cuts is astronomically infeasible.

The latest developments in this continuing story of cake cutting

- After 20 more year, Aziz ans MacKenzie [2016, spring] gave an envy-free protocol for $n = 4$ agents with a bounded number of cuts.
- In the latest development, the same Aziz and McKenzie [2016, fall] gave an envy-free protocol for all n using a bounded number of cuts and queries. However, so far the bound on the number of cuts is astronomically infeasible.

  Namely, the current bound is :

$$n^{n^{n^{n^n}}}$$

  .

Some impossibility results

- Edmonds and Pruhs [2006]: Any proportional cake cutting algorithm require $\Omega(n \log n)$ queries in the Robertson and Web modeli matching (asymptotically) the Evan and Paz discrete version of the moving knife.
- Stromquist [2008]: For $n \geq 3$, no envy free algorithm can produce a contiguous allocation.
- Proccacia [2009]: Any envy free algorithm requires $\Omega(n^2)$ queries in the Robertson and Web model.
- Caragiannis et al [2009]: The *price of proportionality* is $\Theta(\sqrt{n})$ and the price of envy-freeness is at least $\Omega(\sqrt{n})$ and at worst $O(n)$. These concepts are in analogy with the price of anarchy and measure how much fairness costs relative to an optimal allocation in terms of social welfare.

See Proccacia [2016] for a recent survey of such complexity results.

The Selfridge and Conway envy-free protocol for $n = 3$

For a sense of how involved cake cutting procedures can be (even when the number of cuts is small), here is the Selfridge and Conway protocol (using CMU lecture notes by Ariel Proccacia).

- Stage 0
  - 0.1 Agent 1 cuts the cake into three equal parts according to v_1
  - 0.2 Agent 2 "trims" his most valuable piece (according to v_2) of these three pieces so that the trimmed piece Z has the same value as the second most valuable piece M. Lets say that Y is what has been trimmed off and X is the what remains of the entire cake after Y is removed. Note: Z and M are each one of the three pieces in X.
- Stage 1: Dividing X
  - 1.1 Agent 3 chooses a piece of X
  - 1.2 If agent 3 chooses the trimmed piece Z then agent 2 chooses M. Otherwise, if agent 3 chooses L one of the other two pieces in X, then agent 2 chooses Z.
  - 1.3 Agent 1 chooses the remaining piece of X

The last part of the $n = 3$ protocol; dividing the trimmed off part Y

Lets say that agent $i \in \{2, 3\}$ chose Z and agent $j \in \{2, 3\}$ chose L

- Stage 2: Dividing the trimmed off piece Y
  - 2.1 Agent j divides Y into 3 equal pieces according to v_j.
  - 2.2 These three pieces of Y are allocated in the following order: Agent i chooses first, then agent 1, and then agent j.

It is easy to see that the protocol uses 5 cuts if $Y \neq \varnothing$ and 2 cuts if $Y = \varnothing$.

Proving envy-freeness is done by proving that each agent is envy free with respect to their piece in X and with respect to their piece in Y.

Completing the argument for envy-freeness

For the envy-freeness of the division of X, agent 3 chooses first so she is envy-free. Agent 2 gets one of his best two equal valued pieces and agent 1 gets a piece other than Z which has the same value to her.

For the envy-freeness of the division of Y, agent i chooses first so he is envy-free. Agent 1 chooses before agent j so she is not envious of him. Agent 1 is also not envious of agent i because i's share of both X and Y is at most $1/3$ with respect to v_1. Finally, agent j is not envious about his share of Y since he divided Y into three equally valued pieces.