CS364A: Algorithmic Game Theory
Lecture #1: Introduction and Examples*

Tim Roughgarden†

September 23, 2013

1 Mechanism Design: The Science of Rule-Making

This course is roughly organized into three parts, each with its own overarching goal. Here is the first.

Course Goal 1 Understand how to design systems with strategic participants that have good performance guarantees.

We begin with a cautionary tale. In 2012, the Olympics were held in London. One of the biggest scandals of the event concerned, of all sports, women's badminton. The scandal did not involve any failed drug tests, but rather a failed tournament design that did not carefully consider *incentives*.

The tournament design that was used is familiar from World Cup soccer. There are four groups (A,B,C,D) of four teams each. The tournament has two phases. In the first "round-robin" phase, each team plays the other three teams in its group, and does not play teams in other groups. The top two teams from each group advance to the second phase, the bottom two teams from each group are eliminated. In the second phase, the remaining eight teams play a standard "knockout" tournament (as in tennis, for example): there are four quarterfinals (with the losers eliminated), then two semifinals (with the losers playing an extra match to decide the bronze model), and then the final (the winner gets the gold, the loser the silver).

The incentives of participants and of the Olympics committee (and fans) are not necessarily aligned in such a tournament. What does a team want? To get as good a medal as possible, of course. What does the Olympics committee want? They didn't seem to think carefully about this question, but in hindsight it's clear that they want every team to try

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

their best to win every match. Why, you ask, would a team ever want to lose a match? Indeed, in the second "knockout" phase of the tournament, where losing leads to instant elimination, it's obvious that winning is better than losing.

To understand the incentive issues, we need to explain how the eight winners from the round-robin phase are paired up in the quarterfinals. The team with the best record from group A plays the second-best team from group C in the first quarterfinal; similarly with the best team from group C and the second-best time from group A in the third quarterfinal; and similarly with the top two teams from groups B and D in the second and fourth quarterfinals. The dominoes started to fall when, on the last day of round-robin competition, there was a shocking upset: the Danish team of Pedersen and Juhl (PJ) beat the Chinese team of Qing and Wunlei (QW), and as a result PJ won group D with QW coming in second (so both teams advanced to the knockout round).

The first controversial match involved another team from China, Xiaoli and Yang (XY), and the South Korean team of Kyung-eun and Ha-na (KH). Both teams had a 2-0 record in group A play. Thus, both were headed for the knockout stage, with the winner and loser of this match the top and second-best team from the group, respectively. Here was the issue: the group-A winner would meet the fearsome QW team (which came in only second in group D) in the semifinals of the knockout tournament (where a loss means a bronze medal at best), while the the second-best team in group-A would not have to face QW until the final (where a silver medal is guaranteed). Both the XY and KH teams found the difference between these two scenarios significant enough to try to *deliberately lose the match.*[1] This unappealing spectacle led to scandal, derision, and, ultimately, the disqualification of the XY and KH teams, as well as two group C teams (from Indonesia and another team from South Korea) that used the same strategy, for the same reason.[2]

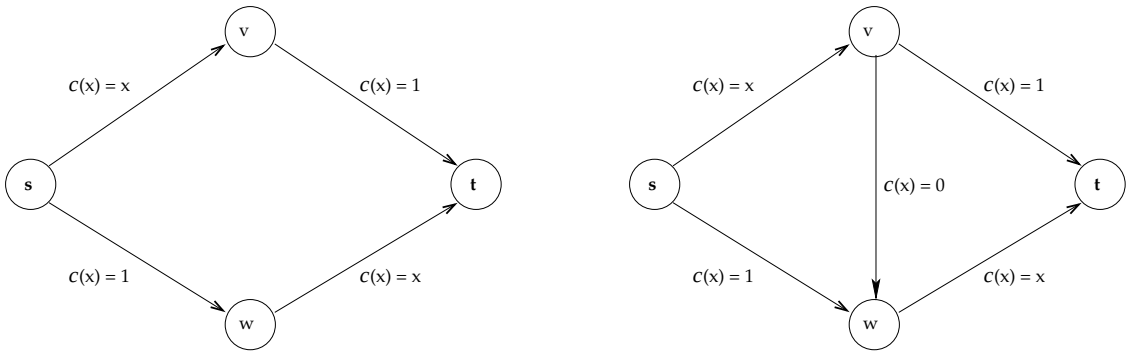
The point is that, in systems with strategic participants, *the rules matter*. Poorly designed systems suffer from unexpected and undesirable results. The burden lies on the system designer to anticipate strategic behavior, not on the participants to behave against their own interests. Can we blame the badminton players for acting to maximize their medal placement? To quote Hartline and Kleinberg [5]:

> "The next time we bemoan people exploiting loopholes to subvert the intent of the rule makers, instead of asking 'What's wrong with these people?" let's instead ask, 'What's wrong with the rules?' and then adopt a scientifically principled approach to fixing them."

There is a well-developed science of rule-making, the field of *mechanism design*. The goal in this field is to design rules so that strategic behavior by participants leads to a desirable outcome. Killer applications of mechanism design, which we will discuss in detail later, include Internet search auctions, wireless spectrum auctions, matching medical residents to hospitals, matching children to schools, and kidney exchange markets.

[1] That the teams feared the Chinese team QW far more than the Danish team PJ seems justified in hindsight: PJ were knocked out in the quarterfinals, while QW won the gold medal.

[2] If you're having trouble imagining what a badminton match looks like when both teams are trying to lose, I encourage you to track down the video on YouTube.



(a) Initial network (b) Augmented network

Figure 1: Braess's Paradox. The addition of an intuitively helpful edge can adversely affect all of the traffic.

We'll study some of the basics of the traditional economic approach to mechanism design, along with several complementary contributions from computer science that focus on computational efficiency, approximate optimality, and robust guarantees.

2 The Price of Anarchy: When Is Selfish Behavior Near-Optimal?

Sometimes you don't have the luxury of designing the rules of a game from scratch, and want instead to understand a game that occurs "in the wild" — the Internet or a road network, for example.

Course Goal 2 When is selfish behavior essentially benign?

2.1 Braess's Paradox

For a motivating example, consider *Braess's Paradox* (Figure 1) [1]. There is a suburb s, a train station t, and a fixed number of drivers who wish to commute from s to t. For the moment, assume two non-interfering routes from s to t, each comprising one long wide road (with travel time one hour, no matter how much traffic uses it) and one short narrow road (with travel time in hours equal to the fraction of traffic using it) as shown in Figure 1(a). The combined travel time in hours of the two edges on one of these routes is $1 + x$, where x is the fraction of the traffic that uses the route. The routes are therefore identical, and traffic should split evenly between them. In this case, all drivers arrive at their destination 90 minutes after their departure from s.

Now, suppose we install a teleportation device allowing drivers to travel instantly from v to w. The new network is shown in Figure 1(b), with the teleporter represented by edge

(v, w) with constant cost $c(x) = 0$, independent of the road congestion. How will the drivers react?

We cannot expect the previous traffic pattern to persist in the new network. The travel time along the new route $s \to v \to w \to t$ is never worse than that along the two original paths, and it is strictly less whenever some traffic fails to use it. We therefore expect all drivers to deviate to the new route. Because of the ensuing heavy congestion on the edges (s, v) and (w, t), all of these drivers now experience two hours of travel time when driving from s to t. Braess's Paradox thus shows that the intuitively helpful action of adding a new zero-cost link can negatively impact *all* of the traffic!

Braess's Paradox shows that selfish routing does not minimize the commute time of the drivers — in the network with the teleportation device, an altruistic dictator could dictate routes to traffic in improve everyone's commute time by 25%. We define the *price of anarchy (POA)* to be the ratio between the system performance with strategic players and the best-possible system performance — for the network in Figure 1(b), the ratio between 2 and $\frac{3}{2}$ (i.e., $\frac{4}{3}$).

The POA was first defined and studied by computer scientists. Every economist and game theorist knows that equilibria are generally inefficient, but until the 21st century there had been almost no attempts to quantify such inefficiency in different application domains.

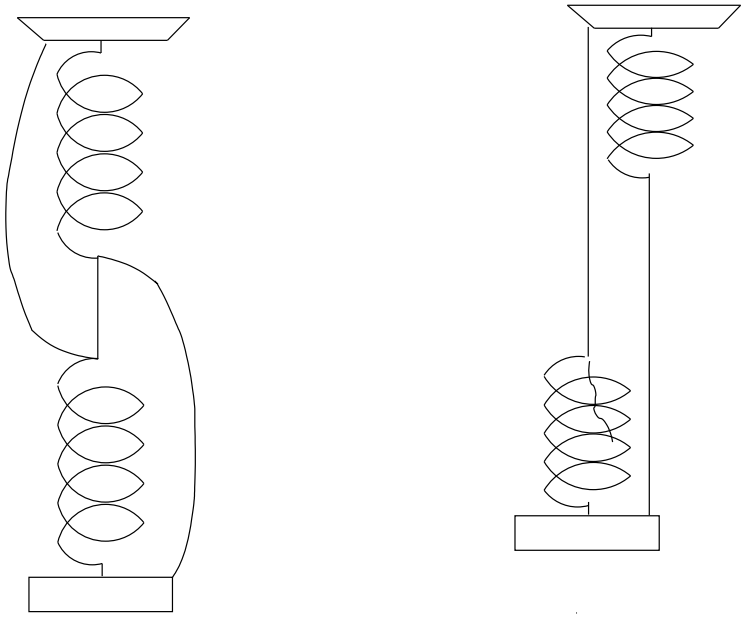
In our study of the POA, the overarching goal is to identify application domains and conditions under which the POA is guaranteed to be close to 1, and thus selfish behavior leads to a near-optimal outcome. Killer applications include network routing, scheduling, resource allocation, and simple auction designs. For example, modest overprovision of network capacity guarantees that the POA of selfish routing is close to 1 [7].

2.2 Strings and Springs

As a final aside, we note that selfish routing is also relevant in systems that have no explicit notion of traffic whatsoever. Cohen and Horowitz [3] gave the following analogue of Braess's Paradox in a mechanical network of strings and springs.

In the device pictured in Figure 2, one end of a spring is attached to a fixed support, and the other end to a string. A second identical spring is hung from the free end of the string and carries a heavy weight. Finally, strings are connected, with some slack, from the support to the upper end of the second spring and from the lower end of the first spring to the weight. Assuming that the springs are ideally elastic, the stretched length of a spring is a linear function of the force applied to it. We can therefore view the network of strings and springs as a traffic network, where force corresponds to traffic and physical distance corresponds to cost.

With a suitable choice of string and spring lengths and spring constants, the equilibrium position of this mechanical network is described by Figure 2(a). Perhaps unbelievably, severing the taut string causes the weight to *rise*, as shown in Figure 2(b)! An explanation for this curiosity is as follows. Initially, the two springs are connected in series, and each bears the full weight and is stretched out to great length. After cutting the taut string, the two springs are only connected in parallel. Each spring then carries only half of the weight, and



(a) Before (b) After

Figure 2: Strings and springs. Severing a taut string lifts a heavy weight.

accordingly is stretched to only half of its previous length. The rise in the weight is the same as the improvement in the selfish outcome obtained by deleting the zero-cost edge of Figure 1(b) to obtain the network of Figure 1(a).

This construction is not merely theoretical; on YouTube you can find several physical demonstrations of Braess's Paradox that were performed (for extra credit) by past students of CS364A.

3 Complexity of Equilibria: How Do Strategic Players Learn?

Some games are easy to play. For example, in the second network of Braess's Paradox (Figure 1(b)), using the teleporter is a no-brainer for every individual — it is the best route, no matter what other drivers do. In many other games, like the first-price auctions mentioned in the next lecture, it's much harder to figure out how to play.

Course Goal 3 How do strategic players reach an equilibrium? (Or do they?)

Informally, an equilibrium is a "steady state" of a system where each participant, assuming everything else stays the same, want to remain as-is. Hopefully, you didn't learn the definition of a Nash equilibrium from the movie *A Beautiful Mind*.

In most games, the best action to play depends on what the other players are doing. Rock-Paper-Scissors, rendered below in "bimatrix" form, is a canonical example.

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

One player chooses a row and the other a column. The numbers in the corresponding matrix entry are the payoffs for the row and column player, respectively.

There is certainly no "determinstic equilibrium" in the Rock-Paper-Scissors game: whatever the current state, at least one player would benefit from a unilateral move.

A crucial idea, developed largely by von Neumann, is to allow *randomized* (a.k.a. *mixed*) strategies. (In a game like Rock-Paper-Scissors, from your perspective, your opponent is effectively randomizing.) If both players randomize uniformly in Rock-Paper-Scissors, then neither player can increase their expected payoff via a unilateral deviation — indeed, every unilateral deviation yields zero expected payoff to the deviator. A pair of probability distributions with this property is a *(mixed-strategy) Nash equilibrium*.

Remarkably, with randomization, *every* game has at least one Nash equilibrium.

Nash's Theorem ('51): Every bimatrix game has a Nash equilibrium.

Nash's theorem holds more generally in games with any finite number of players.

Another piece of good news, that we'll cover in the course: if a bimatrix game is *zero-sum* — meaning that the payoff pair in each entry sums to zero, like in Rock-Paper-Scissors — then a Nash equilibrium can be computed in polynomial time. This can be done via linear programming or, if a small amount of error can be tolerated, via simple iterative learning algorithms. There algorithmic results give credence to the Nash equilibrium concept as a good prediction of behavior in zero-sum games.

Far more recently (mid-last decade), computer science contributed an important negative result: under suitable complexity assumptions, there is *no* polynomial-time for computing a Nash equilibrium in general (non-zero-sum) games [2, 4]. While the problem is not NP-hard (unless $NP = coNP$), it is something called "PPAD-hard", which we will explain and interpret in due course.

This hardness result is interesting for at least two different reasons. On a technical level, it shows that computing Nash equilibria is a fundamental computational problem of "intermediate" difficulty (like factoring and graph isomorphism) — unlikely to be in P or NP-complete. On a conceptual level, many interpretations of an equilibrium concept involve someone — the participants or a designer — determining an equilibrium. For example, the idea that markets implicitly compute a solution to a significant computational problem goes back at least to Adam Smith's notion of the invisible hand [8]. If all parties are boundedly rational, then an equilibrium can be interpreted as a credible prediction only if it can be computed with reasonable effort. Rigorous intractability results thus cast doubt on the predictive power of equilibrium concepts (a critique that dates back at least to Rabin [6]). While intractability is certainly not the first stone thrown at the Nash equilibrium concept, it

is one that theoretical computer science is ideally situated to make precise. This perspective also provides novel motivation for our study of "easier" equilibrium concepts, like correlated equilibria and coarse correlated equilibria.

4 What Computer Science Brings to the Table

There is, of course, a very rich literature on mechanism design and equilibria in economics. In this course, we'll see how computer scientists have complemented this literature in a number of novel ways. The traditional economics approach to the topics we study tends to focus on "Bayesian" (i.e., average-case) analyses; emphasizes exact solutions and characterizations; and usually ignores computational issues. Computer science has offered a focus on and a language to discuss computational complexity; popularized the widespread use of approximation bounds to reason about models where exact solutions are unrealistic or unknowable; and encouraged more robust performance guarantees.

5 Target Audience

These notes assume a background in undergraduate theoretical computer science — basic algorithms and NP-completeness. They do not assume any background in game theory or economics. (Conversely, this course is no substitute for a traditional game theory or microeconomics economics.) The level is meant to be accessible to a Masters or 1st-year PhD student with an affinity for theory.

References

[1] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[2] X. Chen, X. Deng, and S.-H. Teng. Settling the complexity of two-player Nash equilibria. *Journal of the ACM*, 56(3), 2009.

[3] J. E. Cohen and P. Horowitz. Paradoxical behavior of mechanical and electrical networks. *Nature*, 352(8):699–701, 1991.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[5] J. D. Hartline and R. D. Kleinberg. Badminton and the science of rule making. *The Huffington Post*, August 13 2012. `http://www.huffingtonpost.com/jason-hartline/badminton-and-the-science-of-rule-making_b_1773988.html`.

[6] M. O. Rabin. Effective computability of winning strategies. In M. Dresher, A. W. Tucker, and P. Wolfe, editors, *Contributions to the Theory Games*, volume 3, pages 147–157. Princeton, 1957.

[7] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003.

[8] A. Smith. *An Inquiry Into the Nature and Causes of the Wealth of Nations*. Methuen, 1776.

CS364A: Algorithmic Game Theory
Lecture #2: Mechanism Design Basics*

Tim Roughgarden†

September 25, 2013

1 Single-Item Auctions

The most sensible place to start our discussion of mechanism design — the science of rule-making — is *single-item auctions*. Recall our overarching goal in this part of the course.

Course Goal 1 Understand how to design systems with strategic participants that have good performance guarantees.

Consider a seller that had a single good, such as a slightly antiquated smartphone. This is the setup in a typical eBay auction, for example. There is some number n of (strategic!) bidders who are potentially interested in buying the item.

We want to reason about bidder behavior in various auction formats. To do this, we need a model of what a bidder wants. The first key assumption is that each bidder i has a *valuation* v_i — its maximum willingness-to-pay for the item being sold. Thus bidder i wants to acquire the item as cheaply as possible, provided the selling price is at most v_i. Another important assumption is that this valuation is *private*, meaning it is unknown to the seller and to the other bidders.

Our bidder utility model, called the *quasilinear utility model*, is as follows. If a bidder loses an auction, its utility is 0. If the bidder wins at a price p, then its utility is $v_i - p$. This is perhaps the simplest natural utility model, and it is the one we will focus on in this course.[1]

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]More complex utility models are well motivated and have been studied — to model risk attitudes, for example.

2 Sealed-Bid Auctions

For now, we'll focus on a particular simple class of auction formats: *sealed-bid auctions*. Here's what happens:

(1) Each bidder i privately communicates a bid b_i to the auctioneer — in a sealed envelope, if you like.

(2) The auctioneer decides who gets the good (if anyone).

(3) The auctioneer decides on a selling price.

There is an obvious way to implement step (2) — give the good to the highest bidder. Today, this will be the only selection rule that we study.[2]

There are multiple reasonable ways to implement step (3) and the implementation significantly affects bidder behavior. For example, suppose we try to be altruistic and charge the winning bidder nothing. This idea backfires badly, with the auction devolving into a game of "who can name the highest number"?

3 First-Price Auctions

A much more reasonable choice is to ask the winning bidder to pay its bid. This is called a *first-price auction*, and such auction are common in practice.

First-price auctions are hard to reason about. First, as a participant, it's hard to figure out how to bid. Second, as a seller or auction designer, it's hard to predict what will happen. We'll elaborate on the theory of first-price auctions in Problem Set #1 and in later advanced material. For now, to drive the point home, imagine participating in the following experiment. There's an item being sold via a first-price auction. Your valuation (in dollars) is the number of your birth month plus the day of your birth. Thus, your valuation is somewhere between 2 (for January 1st) and 43 (for December 31st). Suppose there is exactly one other bidder (drawn at random from the world) whose valuation is determined in the same way. What bid would you submit to maximize your (quasilinear) utility? Would your answer change if you knew there were two other bidders in the auction, rather than one?

4 Second-Price Auctions

Let's now focus on a different auction format, which is also common in practice, that is much easier to reason about. To motivate it, think about what happens when you win an eBay auction. If you bid \$100 and win, do you necessarily pay \$100? Not necessarily — eBay uses a "proxy bidder" that increases your bid on your behalf until your maximum bid

[2]When we study revenue maximization a few lectures hence, we'll see why other winner selection rules are important.

is reached, or until you are the highest bidder (whichever comes first). For example, if the highest other bid is only \$90, then you will only pay \$90 (plus a small increment), rather than your maximum bid \$100. The upshot is: *if you win an eBay auction, the sale price equals the highest other bid (the second highest overall), plus a small increment.*

A *second-price* or *Vickrey* auction is a sealed-bid auction in which the highest bidder wins and pays a price equal to the second-highest bid.

Claim 4.1 *In a second-price auction, every bidder has a dominant strategy: set its bid b_i equal to its private valuation v_i. That is, this strategy maximizes the utility of bidder i, no matter what the other bidders do.*

This claim implies that second-price auctions are particularly easy to participate in — you don't need to reason about the other bidders in any way (how many there are, what their valuations, whether or not they bid truthfully, etc.) to figure out how you should bid. Note this is completely different from a first-price auction. You should never bid your valuation in a first-price auction (that would guarantee zero utility), and the ideal amount to underbid depends on the bids of the other players.

Proof of Claim 4.1: Fix an arbitrary player i, its valuation v_i, and the bids $\mathbf{b}_{-i}$ of the other players. (Here $\mathbf{b}_{-i}$ means the vector $\mathbf{b}$ of all bids, but with the ith component deleted. It's wonky notation but you need to get used to it.) We need to show that bidder i's utility is maximized by setting $b_i = v_i$. (Recall v_i is i's immutable valuation, while it can set its bid b_i to whatever it wants.)

Let $B = \max_{j \neq i} b_j$ denote the highest bid by some other bidder. What's special about a second-price auction is that, even though there are an infinite number of bids that i could make, only distinct outcomes can result. If $b_i < B$, then i loses and receives utility 0. If $b_i \geq B$, then i wins at price B and receives utility $v_i - B$.[3]

We now consider two cases. First, if $v_i < B$, the highest utility that bidder i can get is $\max\{0, v_i - B\} = 0$, and it achieves this by bidding truthfully (and losing). Second, if $v_i \geq B$, the highest utility that bidder i can get is $\max\{0, v_i - B\} = v_i - B$, and it achieves this by bidding truthfully (and winning). ■

The second important property is that a truthtelling bidder will never regret participating in a second-price auction.

Claim 4.2 *In a second-price auction, every truthtelling bidder is guaranteed non-negative utility.*

Proof: Losers all get utility 0. If bidder i is the winner, then its utility is $v_i - p$, where p is the second-highest bid. Since i is winner (and hence the highest bidder) and bid its true valuation, $p \leq v_i$ and hence $v_i - p \geq 0$. ■

[3] We're assuming here that ties are broken in favor of bidder i. The claim holds no matter how ties are broken, as you should check.

The exercises ask you to explore further properties of and variations on the Vickrey auction. For example, truthful bidding is the *unique* dominant strategy for a bidder in a Vickrey auction.

5 Awesome Auctions

Taking a step back, we can claim the following.

Theorem 5.1 (Vickrey [3]) *The Vickrey auction is awesome. Meaning, it enjoys three quite different and desirable properties:*

(1) [**strong incentive guarantees**] *It is* dominant-strategy incentive-compatible (DSIC)*, i.e., Claims 4.1 and 4.2 hold.*

(2) [**strong performance guarantees**] *If bidders report truthfully, then the auction maximizes the* social surplus

$$\sum_{i=1}^{n} v_i x_i, \tag{1}$$

where x_i is 1 if i wins and 0 if i loses, subject to the obvious feasibility constraint that $\sum_{i=1}^{n} x_i \leq 1$ (i.e., there is only one item).[4]

(3) [**computational efficiency**] *The auction can be implemented in polynomial (indeed, linear) time.*

All of these properties are important. From a bidder's perspective, the DSIC property, which guarantees that truthful reporting is a dominant strategy and never leads to negative utility, makes it particularly easy to choose a bid. From the seller's or auction designer's perspective, the DSIC property makes it much easier to reason about the auction's outcome. Note that *any* prediction of an auction's outcome has to be predicated on assumptions about how bidders behave. In a DSIC auction, one only has to assume that a bidder with an obvious dominant strategy will play it — behavioral assumptions don't get much weaker than that.

The DSIC property is great when you can get it, but we also want more. For example, an auction that gives the item away for free to a random bidder is DSIC, but it makes no effort to identify which bidders actually want the good. The surplus-maximization property states something rather amazing: even though the bidder valuations were a priori unknown to the auctioneer, the auction nevertheless successfully identifies the bidder with the highest valuation! (Assuming truthful bids, which is a reasonable assumption in light of the DSIC property.) That is, the Vickrey auction solves the surplus-maximization optimization problem as well as if the data (the valuations) were known in advance.

[4]Note that the sale price is not part of the surplus. The reason is that we treat the auctioneer as a player whose utility is the revenue it earns; its utility then cancels out the utility lost by the auction winner from paying for the item. We will discuss auctions for maximizing seller revenue in a few lectures.

The importance of the third property is self-evident to computer scientists. To have potential practical utility, an auction should run in a reasonable amount of time — or even in real time, for some applications. Auctions with super-polynomial running time are useful only for fairly small instances.

The next several lectures strive for awesome auctions, in the sense of Theorem 5.1, for applications beyond single-item auctions. The two directions we focus on are well motivated: more complex allocation problems, like those that inevitably arise in sponsored search and combinatorial auctions; and maximizing seller revenue in lieu of social surplus.

6 Case Study: Sponsored Search Auctions

6.1 Background

A Web search results page comprises a list of organic search results — deemed by some underlying algorithm, such as PageRank, to be relevant to your query — and a list of sponsored links, which have been paid for by advertisers. (Go do a Web search now to remind yourself, preferably on a valuable keyword like "mortgage" or "asbestos".) Every time you type a search query into a search engine, an auction is run in real time to decide which advertisers' links are shown, in what order, and how they are charged. It is impossible to overstate how important such *sponsored search auctions* have been to the Internet economy. Here's one jaw-dropping statistic: around 2006, sponsored auctions generate roughly 98% of Google's revenue [1]. While online advertising is now sold in many different ways, sponsored search auctions continue to generate tens of billions of dollars of revenue every year.

6.2 The Basic Model of Sponsored Search Auctions

We discuss next a simplistic but useful and influential model of sponsored search auctions, due independently to Edelman et al. [1] and Varian [2]. The goods for sale are the k "slots" for sponsored links on a search results page. The bidders are the advertisers who have a standing bid on the keyword that was searched on. For example, Volvo and Subaru might be bidders on the keyword "station wagon," while Nikon and Canon might be bidders on the keyword "camera." Such auctions are more complex than single-item auctions in two ways. First, there are generally multiple goods for sale (i.e., $k > 1$). Second, these goods are *not* identical — slots higher on the search page are more valuable than lower ones, since people generally scan the page from top to bottom.

We quantify the difference between different slots using *click-through-rates (CTRs)*. The CTR α_j of a slot j represents the probability that the end user clicks on this slot. Ordering the slots from top to bottom, we make the reasonable assumption that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$. For simplicity, we also make the unreasonable assumption that the CTR of a slot is independent of its occupant. The exercises show how to extend the following results to the more general and realistic model in which each advertiser i has a "quality score" β_i (the higher the better) and the CTR of advertiser i in slot j is the product $\beta_i \alpha_j$.

We assume that an advertiser is not interested in an impression (i.e., being displayed on a page) per se, but rather has a private valuation v_i for each *click* on its link. Hence, the value derived by advertiser i from slot j is $v_i\alpha_j$.

6.3 What We Want

Is there an awesome sponsored search auction? Our desiderata are:

(1) DSIC. That is, truthful bidding should be a dominant strategy, and never leads to negative utility.

(2) Social surplus maximization. That is, the assignment of bidders to slots should maximize $\sum_{i=1}^{n} v_i x_i$, where x_i now denotes the CTR of the slot to which i is assigned (or 0 if i is not assigned to a slot). Each slot can only be assigned to one bidder, and each bidder gets only one slot.

(3) Polynomial running time. Remember zillions of these auctions need to be run every day!

6.4 Our Design Approach

What's hard about mechanism design problems is that we have to jointly design two things: the choice of who wins what, and the choice of who pays what. Even in single-item auctions, it is not enough to make the "correct" choice to first design decision (i.e., giving the good to the highest bidder) — if the payments are not just right, then strategic participants will game the system.

Happily, in many applications including sponsored search auctions, we can tackle this two-prong design problem one step at a time.

Step 1: Assume, without justification, that bidders bid truthfully. Then, how should we assign bidders to slots so that the above properties (2) and (3) hold?

Step 2: Given our answer to Step 1, how should we set selling prices so that the above property (1) holds?

If we successfully answer both these questions, then we have constructed an awesome auction. Step 2 ensures the DSIC property, which means that bidders will bid truthfully (assuming as usual that a bidder with an obvious dominant strategy does indeed play that strategy). This means that the hypothesis in Step 1 is satisfied, and so the final outcome of the auction is indeed surplus-maximizing (and is computed in polynomial time).

We conclude this lecture by executing Step 1 of sponsored search auctions. Given truthful bids, how should we assign bidders to slots to maximize the surplus? As an exercise, you should show that the natural greedy algorithm is optimal (and runs in near-linear time): assign the jth highest bidder to the jth highest slot for $j = 1, 2, \ldots, k$.

Can we implement Step 2? Is there an analog of the second-price rule — sale prices that render truthful bidding a dominant strategy for every bidder? Next lecture we'll derive an affirmative answer via Myerson's Lemma, a powerful tool in mechanism design.

References

[1] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the Generalized Second-Price Auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[2] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

[3] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

CS364A: Algorithmic Game Theory
Lecture #3: Myerson's Lemma*

Tim Roughgarden†

September 30, 2013

1 The Story So Far

Last time, we introduced the Vickrey auction and proved that it enjoys three desirable and different guarantees:

(1) [**strong incentive guarantees**] DSIC. That is, truthful bidding should be a dominant strategy (and never leads to negative utility).

Don't forget the two reasons we're after the DSIC guarantee. First, such an auction is easy to play for bidders — just play the obvious dominant strategy. Second, assuming only that bidders will play a dominant strategy when they have one, we can confidently predict the outcome of the auction.

(2) [**strong performance guarantees**] Social surplus maximization. That is, assuming truthful bids (which is justified by (1)), the allocation of goods to bidders should maximize $\sum_{i=1}^{n} v_i x_i$, where x_i the amount of stuff allocated to bidder i.

(3) [**computational efficiency**] The auction can be implemented in polynomial (indeed, linear) time.

To extend these guarantees beyond single-item auctions to more complex problems, like the sponsored search auctions introduced last lecture, we advocated a two-step design approach.

Step 1: Assume, without justification, that bidders bid truthfully. Then, how should we assign bidders to slots so that the above properties (2) and (3) hold?

Step 2: Given our answer to Step 1, how should we set selling prices so that the above property (1) holds?

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

For instance, in sponsored search auctions, the first step can be implemented using a simple greedy algorithm (assign the jth highest bidder the jth best slot). But what about the second step?

This lecture states and proves *Myerson's Lemma*, a powerful and general tool for implementing Step 2. This lemma applies to sponsored search auctions as a special case, and we'll also see further applications later.

2 Single-Parameter Environments

A good level of abstraction at which to state Myerson's Lemma is *single-parameter environments*. Such an environment has some number n of bidders. Each bidder i has a private valuation v_i, its value "per unit of stuff" that it gets. Finally, there is a *feasible set* X. Each element of X is an n-vector $(x_1, x_2, \ldots, x_n)$, where x_i denotes the "amount of stuff" given to bidder i. For example:

- In a single-item auction, X is the set of 0-1 vectors that have at most one 1 (i.e., $\sum_{i=1}^{n} x_i \leq 1$).
- With k identical goods and the constraint the each customer gets at most one, the feasible set is the 0-1 vectors satisfying $\sum_{i=1}^{n} x_i \leq k$.
- In sponsored search, X is the set of n-vectors corresponding to assignments of bidders to slots, where each slot is assigned at most one bidder and each bidder is assigned at most one slot. If bidder i is assigned to slot j, then the component x_i equals the CTR α_j of its slot.

3 Allocation and Payment Rules

Recall that a sealed-bid auction has to make two choices: who wins and who pays what. These two decisions are formalized via an *allocation rule* and a *payment rule*, respectively. That is, a sealed-bid auction has three steps:

(1) Collect bids $\mathbf{b} = (b_1, \ldots, b_n)$

(2) [**allocation rule**] Choose a feasible allocation $\mathbf{x}(\mathbf{b}) \in X \subseteq \mathcal{R}^n$ as a function of the bids.

(3) [**payment rule**] Choose payments $\mathbf{p}(\mathbf{b}) \in \mathcal{R}^n$ as a function of the bids.

We continue to use a quasilinear utility model, so, in an auction with allocation and payment rules $\mathbf{x}$ and $\mathbf{p}$, respectively, bidder i has utility

$$u_i(\mathbf{b}) = v_i \cdot x_i(\mathbf{b}) - p_i(\mathbf{b})$$

on the bid profile (i.e., bid vector) $\mathbf{b}$.

In lecture, we will focus on payment rules that satisfy

$$p_i(\mathbf{b}) \in [0, b_i \cdot x_i(\mathbf{b})] \tag{1}$$

for every i and $\mathbf{b}$. The constraint that $p_i(\mathbf{b}) \geq 0$ is equivalent to prohibiting the seller from paying the bidders. The constraint that $p_i(\mathbf{b}) \leq b_i \cdot x_i(\mathbf{b})$ ensures that a truthtelling bidder receives nonnegative utility (do you see why?).

There are applications where it makes sense to relax one or both of these restrictions on payments, but we won't cover any in these lectures.

4 Statement of Myerson's Lemma

We now come to two important definitions. Both articulate a property of allocation rules.

Definition 4.1 (Implementable Allocation Rule) An allocation rule $\mathbf{x}$ for a single-parameter environment is *implementable* if there is a payment rule $\mathbf{p}$ such the sealed-bid auction $(\mathbf{x}, \mathbf{p})$ is DSIC.

That is, the implementable allocation rules are those that extend to DSIC mechanisms. Equivalently, the projection of DSIC mechanisms onto their allocation rules is the set of implementable rules. If our aim is to design a DSIC mechanism, we must confine ourselves to implementable allocation rules — they form our "design space." In this terminology, we can rephrase the cliffhanger from the end of last lecture as: is the surplus-maximizing allocation rule for sponsored search, which assigns the jth highest bidder to the jth best slot, implementable?

For instance, consider a single-item auction. Is the allocation rule that awards the good to the highest bidder implementable? Sure — we've already constructed a payment rule, the second-price rule, that renders it DSIC. What about the allocation rule that awards the good to the *second-highest* bidder? Here, the answer is not clear: we haven't seen a payment rule that extends it to a DSIC mechanism, but it also seems tricky to argue that no payment rule could conceivably work.

Definition 4.2 (Monotone Allocation Rule) An allocation rule $\mathbf{x}$ for a single-parameter environment is *monotone* if for every bidder i and bids $\mathbf{b}_{-i}$ by the other bidders, the allocation $x_i(z, \mathbf{b}_{-i})$ to i is nondecreasing in its bid z.

That is, in a monotone allocation rule, bidding higher can only get you more stuff.

For example, the single-item auction allocation rule that awards the good to the highest bidder is monotone: if you're the winner and you raise your bid (keeping other bids constant), you continue to win. By contrast, awarding the good to the second-highest bidder is a non-monotone allocation rule: if you're the winner and raise your bid high enough, you lose.

The surplus-maximizing allocation rule for sponsored search, with the jth highest bidder awarded the jth slot, is monotone. The reason is that raising your bid can only increase

your position in the sorted order of bids, which can only net you a higher slot, which can only increase the click-through-rate of your slot.

We state Myerson's Lemma in three parts; each is conceptually interesting and will be useful in later applications.

Theorem 4.3 (Myerson's Lemma [2]) *Fix a single-parameter environment.*

(a) An allocation rule $\mathbf{x}$ *is implementable if and only if it is monotone.*

(b) If $\mathbf{x}$ *is monotone, then there is a unique payment rule such that the sealed-bid mechanism* $(\mathbf{x}, \mathbf{p})$ *is DSIC [assuming the normalization that* $b_i = 0$ *implies* $p_i(\mathbf{b}) = 0$*].*

(c) The payment rule in (b) is given by an explicit formula (see (6)*, below).*

Myerson's Lemma is the foundation on which we'll build most of our mechanism design theory. Part (a) states that Definitions 4.1 and 4.2 define exactly the same class of allocation rules. This equivalence is incredibly powerful: Definition 4.1 describes our design goal but is unwieldy to work with and verify, while Definition 4.2 is far more "operational." Usually, it's not difficult to check whether or not an allocation rule is monotone. Part (b) states that, when an allocation rule is implementable, there is no ambiguity in how to assign payments to achieve the DSIC property — there is only one way to do it. (Assuming bidding zero guarantees zero payment; note this follows from our standing assumption (1).) Moreover, there is a relatively simple and explicit formula for this payment rule (part (c)), a property we apply to sponsored search auctions below and to revenue-maximization auction design in future lectures.

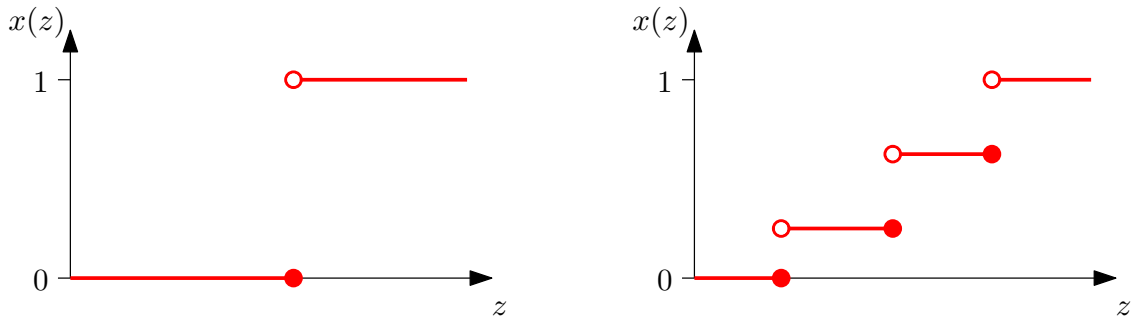
5 Proof of Myerson's Lemma (Theorem 4.3)

Consider an allocation rule $\mathbf{x}$, which may or may not be monotone. Suppose there is a payment rule $\mathbf{p}$ such that $(\mathbf{x}, \mathbf{p})$ is a DSIC mechanism — what could $\mathbf{p}$ look like? The plan of this proof is to cleverly invoke the stringent DSIC constraint to whittle the possibilities for $\mathbf{p}$ down to a single candidate. We will establish all three parts of the theorem in one fell swoop.

Recall the DSIC condition: for every bidder i, every possible private valuation b_i, every set of bids $\mathbf{b}_{-i}$ by the other players, it must be that i's utility is maximized by bidding truthfully. For now, fix i and $\mathbf{b}_{-i}$ arbitrarily. As shorthand, write $x(z)$ and $p(z)$ for the allocation $x_i(z, \mathbf{b}_{-i})$ and payment $p_i(z, \mathbf{b}_{-i})$ of i when it bids z, respectively. Figure 1 gives two examples of what the function x might look like.

We invoke the DSIC constraint via a simple but clever swapping trick. Suppose $(\mathbf{x}, \mathbf{p})$ is DSIC, and consider any $0 \leq y < z$. Because bidder i might well have private valuation z and can submit the false bid y if it wants, DSIC demands that

$$\underbrace{z \cdot x(z) - p(z)}_{\text{utility of bidding } z} \geq \underbrace{z \cdot x(y) - p(y)}_{\text{utility of bidding } y} \tag{2}$$



(a) 0-1 Monotone Curve (b) Piecewise Constant Monotone Curve

Figure 1: Examples of allocation curves $x(\cdot)$.

Similarly, since bidder i might well have the private valuation y and could submit the false bid z, $(\mathbf{x}, \mathbf{p})$ must satisfy

$$\underbrace{y \cdot x(y) - p(y)}_{\text{utility of bidding } y} \geq \underbrace{y \cdot x(z) - p(z)}_{\text{utility of bidding } z} \tag{3}$$

Myerson's Lemma is, in effect, trying to solve for the payment rule $\mathbf{p}$ given the allocation rule $\mathbf{x}$. Rearranging inequalities (2) and (3) yields the following "payment difference sandwich," bounding $p(y) - p(z)$ from below and above:

$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)] \tag{4}$$

The payment difference sandwich already implies one major component of Myerson's Lemma — do you see why?

Thus, we can assume for the rest of the proof that $\mathbf{x}$ is monotone. We will be slightly informal in the following argument, but will cover all of the major ideas of the proof.

In (4), fix z and let y tends to z from above. We focus primarily on the case where x is piecewise constant, as in Figure 1. In this case, x is flat except for a finite number of "jumps". Taking the limit $y \downarrow z$ in (4), the left- and right-hand sides become 0 if there is no jump in x at z. If there is a jump of magnitude h at z, then the left- and right-hand sides both tend to $z \cdot h$. This implies the following constraint on p, for every z:

$$\text{jump in } p \text{ at } z = z \cdot \text{jump in } x \text{ at } z \tag{5}$$

Thus, assuming the normalization $p(0) = 0$, we've derived the following *payment formula*, for every bidder i, bids $\mathbf{b}_{-i}$ by other bidders, and bid b_i by i:

$$p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \text{jump in } x_i(\cdot, \mathbf{b}_{-i}) \text{ at } z_j, \tag{6}$$

where $z_1, \ldots, z_\ell$ are the breakpoints of the allocation function $x_i(\cdot, \mathbf{b}_{-i})$ in the range $[0, b_i]$

A similar argument applies when x is a monotone function that is not necessarily piecewise constant. For instance, suppose that x is differentiable. Dividing the payment difference sandwich (4) by $y - z$ and taking the limit as $y \downarrow z$ yields the constraint

$$p'(z) = z \cdot x'(z)$$

and, assuming $p(0) = 0$, the payment formula

$$p_i(b_i, \mathbf{b}_{-i}) = \int_0^{b_i} z \cdot \tfrac{d}{dz} x_i(z, \mathbf{b}_{-i}) dz \tag{7}$$

for every bidder i, bid b_i, and bids $\mathbf{b}_{-i}$ by the others.

We reiterate that the payment formula in (6) is the *only* payment rule with a chance of extending the given piecewise constant allocation rule $\mathbf{x}$ into a DSIC mechanism. Thus, for every allocation rule $\mathbf{x}$, there is at most one payment rule $\mathbf{p}$ such that $(\mathbf{x}, \mathbf{p})$ is DSIC (cf., part (b) of Theorem 4.3). But the proof is not complete — we still have to check that this payment rule works provided $\mathbf{x}$ is monotone! Indeed, we already know that even this payment rule fails when $\mathbf{x}$ is not monotone.

We give a proof by picture that, when $\mathbf{x}$ is monotone and piecewise constant and $\mathbf{p}$ is defined by (6), then $(\mathbf{x}, \mathbf{p})$ is a DSIC mechanism. The same argument works more generally for monotone allocation rules that are not piecewise constant, with payments defined as in (7). This will complete the proof of all three parts of Myerson's Lemma.

Figures 2(a)–(i) depict the utility of a bidder when it bids truthfully, overbids, and underbids, respectively. The allocation curve $x(z)$ and the private valuation v of the bidder is the same in all three cases. Recall that the bidder's utility when it bids b is $v \cdot x(b) - p(b)$. We depict the first term $v \cdot x(b)$ as a shaded rectangle of width v and height $x(b)$ (Figures 2(a)–(c)). Using the formula (6), we see that the payment $p(b)$ can be represented as the shaded area to the left of the allocation curve in the range $[0, b]$ (Figures 2(d)-(f)). The bidder's utility is the difference between these two terms (Figures 2(g)-(i)). When the bidder bids truthfully, its utility is precisely the area under the allocation curve in the range $[0, v]$ (Figure 2(g)).[1] When the bidder overbids, its utility is this same area, minus the area above the allocation curve in the range $[v, b]$ (Figure 2(h)). When the bidder underbids, its utility is a subset of the area under the allocation curve in the range $[0, v]$ (Figure 2(i)). Since the bidder's utility is the largest in the first case, the proof is complete.

6 Applying the Payment Formula: Sponsored Search Solved

Myerson's payment formula (6) is easy to understand and apply in many applications. For starters, consider a single-item auction with the allocation rule that allocates the good to

[1] In this case, the social surplus contributed by this bidder ($v \cdot x(v)$) naturally splits into its utility (or "consumer surplus"), the area under the curve, and the seller revenue, the area above the curve (in the range $[0, v]$).

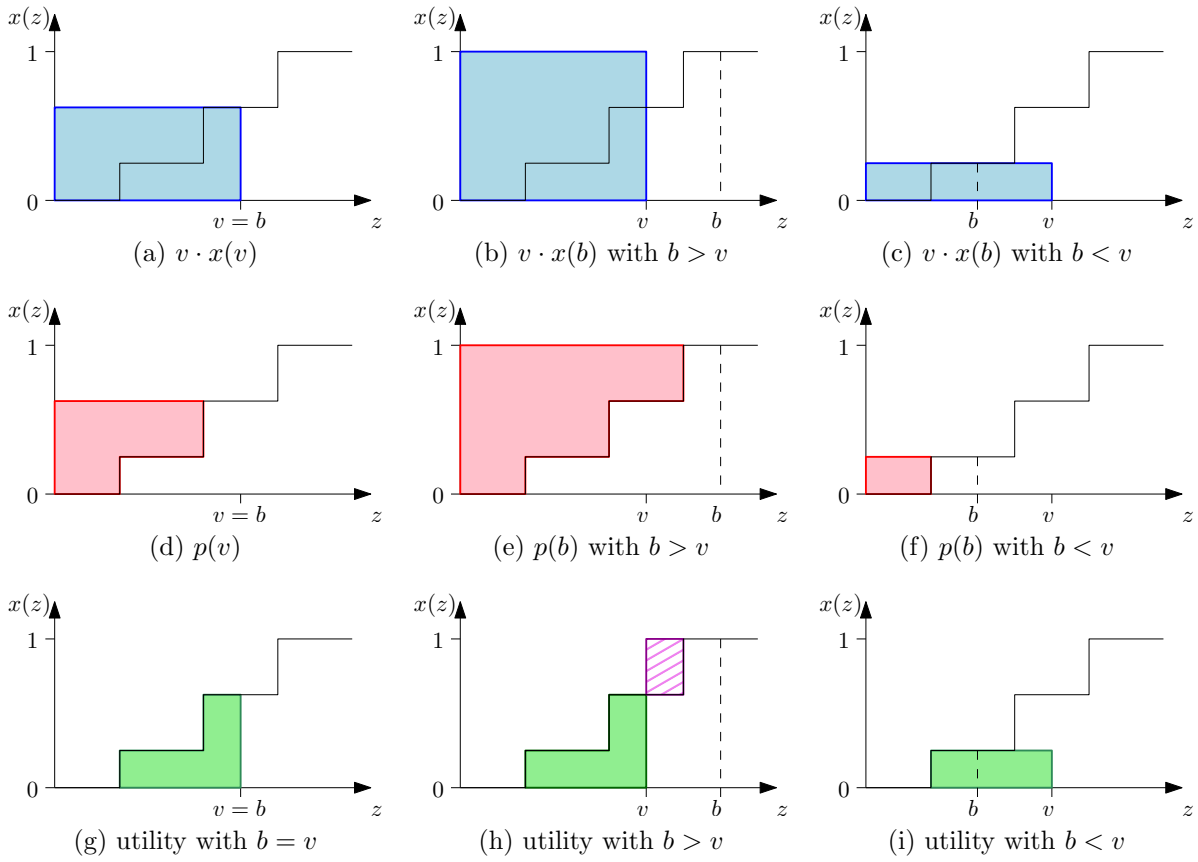


Figure 2: Proof by picture that the payment rule in (6), coupled with the given monotone and piecewise constant allocation rule, yields a DSIC mechanism. The three columns consider the cases of truthful bidding, overbidding, and underbidding, respectively. The three rows show the surplus $v \cdot x(b)$, the payment $p(b)$, and the utility $v \cdot x(b) - p(b)$, respectively. In (h), the solid region represents positive utility and the lined region represents negative utility.

the highest bidder. Fixing i and $\mathbf{b}_{-i}$, the function $x_i(\cdot, \mathbf{b}_{-i})$ is 0 up to $B = \max_{j \neq i} b_j$ and 1 thereafter. The formula (6) is either 0 (if $b_i < B$) or, if $b_i > B$, there is a single breakpoint (a jump of 1 at B) and the payment is $p_i(b_i, \mathbf{b}_{-i}) = B$. Thus, Myerson's Lemma regenerates the Vickrey auction as a special case.

Now let's return to sponsored search auctions. Recall from last lecture that we have k slots with click-through-rates (CTRs) $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$. Let $\mathbf{x}(\mathbf{b})$ be the allocation rule that assigns the jth highest bidder to the jth highest slot, for $j = 1, 2, \ldots, k$. We've noted previously that this rule is surplus-maximizing (assuming truthful bids) and monotone. Applying Myerson's Lemma, we can derive a unique payment rule $\mathbf{p}$ such that $(\mathbf{x}, \mathbf{p})$ is DSIC. To describe it, consider a bid profile $\mathbf{b}$; we can re-index the bidders so that $b_1 \geq b_2 \geq \cdots \geq b_n$. For intuition, focus on the first bidder and imagine increasing its bid from 0 to b_1, holding the other bids fixed. The allocation $x_i(z, \mathbf{b}_{-i})$ ranges from 0 to α_1 as z ranges from 0 to b_1, with a jump of $\alpha_j - \alpha_{j+1}$ at the point where z becomes the jth highest bid in the profile $(z, \mathbf{b}_{-i})$, namely b_{j+1}. Thus, in general, Myerson's payment formula specializes to

$$p_i(\mathbf{b}) = \sum_{j=i}^{k} b_{j+1}(\alpha_j - \alpha_{j+1}) \tag{8}$$

for the ith highest bidder (where $\alpha_{k+1} = 0$).

Recall our assumption that bidders don't care about impressions (i.e., having their link shown), except inasmuch as it leads to a click. This motivates charging bidders per click, rather than per impression. The per-click payment for bidder/slot i is simply that in (8), scaled by $\frac{1}{\alpha_i}$:

$$p_i(\mathbf{b}) = \sum_{j=i}^{k} b_{j+1} \frac{\alpha_j - \alpha_{j+1}}{\alpha_i}. \tag{9}$$

Observe that the formula in (9) has the pleasing interpretation that, when its link its clicked, an advertiser pays a suitable convex combination of the lower bids.

By historical accident, the sponsored search auctions used in real life are based on the "Generalized Second Price (GSP)" auction [1, 3], which is a simpler (and perhaps incorrectly implemented) version of the DSIC auction above. The per-click payment in GSP is simply the next lowest bid. Since Myerson's Lemma implies that the payment rule in (9) is the unique one that yields the DSIC property, we can immediately conclude that the GSP auction is *not* DSIC. It still has a number of nice properties, however, and is "partially equivalent" to the DSIC auction in a precise sense. The Problems ask you to explore this equivalence in detail.

References

[1] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the Generalized Second-Price Auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[2] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[3] H. R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

CS364A: Algorithmic Game Theory
Lecture #4: Algorithmic Mechanism Design*

Tim Roughgarden†

October 2, 2013

1 Knapsack Auctions

Next we design DSIC mechanisms for *knapsack auctions*. These will be single-parameter environments, so Myerson's Lemma will apply.

1.1 Problem Definition

In a knapsack auction, each bidder i has a publicly known *size* w_i (e.g., the duration of a TV ad) and a private valuation (e.g., a company's willingness-to-pay for its ad being shown during the Super Bowl). The seller has a capacity W (e.g., the length of a commercial break). The feasible set X is defined as the 0-1 n-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.) Other situations such knapsack auctions model 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. (When there is a shared resource with limited capacity, you have a Knapsack problem.) Note that k-item auctions (k identical copies of a good, one per customer) is the special case where $w_i = 1$ for all i and $W = k$. Here, different bidders can have different sizes.

Let's try to design an awesome auction using our two-step design paradigm. Recall that we first assume without justification that bids equal values and then decide on our allocation rule. Then we pay the piper and devise a payment rule that extends the allocation rule to a DSIC mechanism.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

1.2 A Surplus-Maximizing DSIC Mechanism

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

$$\mathbf{x}(\mathbf{b}) = \arg\max_{X} \sum_{i=1}^{n} b_i x_i. \tag{1}$$

That is, the allocation rule solves an instance of the Knapsack problem[1] in which the item (i.e., bidder) values are the given 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 surplus.

1.3 Critical Bids

Myerson's Lemma (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 bids $\mathbf{b}_{-i}$ by the other bidders. Since the allocation rule is monotone and 0-1, the allocation curve $x_i(\cdot, \mathbf{b}_{-i})$ is extremely simple: it is 0 until some breakpoint z, at which point it jumps to 1 (Figure 1). Recall the payment formula

$$p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \text{jump in } x_i(\cdot, \mathbf{b}_{-i}) \text{ at } z_j, \tag{2}$$

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, it loses and pays 0. If i bids more than z, it pays $z \cdot (1 - 0) = z$. That is, i pays its *critical bid* — the lowest bid it could make and continue to win (holding the other bids $\mathbf{b}_{-i}$ fixed). Note this is exactly what's going on in the Vickrey auction.

1.4 Intractability of Surplus Maximization

The mechanism proposed in Section 1.2 maximizes social surplus, assuming truthful bids — an assumption justified by its DSIC property. The mechanism thus solves the surplus-maximization problem with unknown data (the v_i's) as well as if this data was known a priori. But is the mechanism awesome in the sense of the Vickrey auction (Lecture 2)? Recall this means:

(1) DSIC.

(2) Surplus-maximizing, assuming truthful bids.

[1] An instance of the Knapsack problem is defined by $2n + 1$ 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. See any undergraduate textbook for more details.

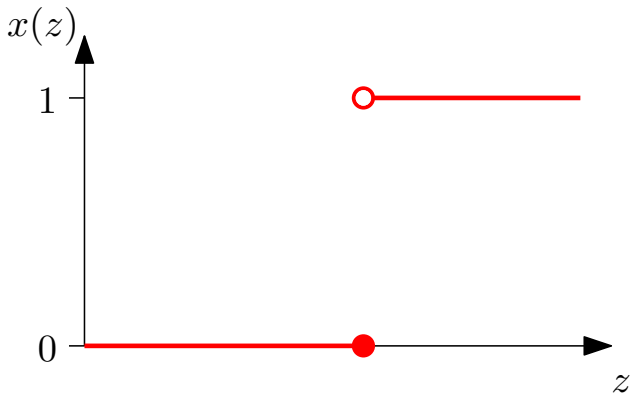


Figure 1: A monotone 0-1 allocation rule.

(3) Runs in polynomial time.

The answer is *no*. The reason is that the Knapsack problem is NP-hard. Thus, there is no polynomial-time implementation of the allocation rule in (1), unless $P = NP$. Thus, properties (2) and (3) are incompatible.

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

A perfectly valid 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 (1) can be implemented in pseudopolynomial time using dynamic programming (again, see any undergraduate algorithms textbook for details). More generally, if your instances are small or structured enough and you have enough time and computing power to implement optimal surplus-maximization, by all means do so — the resulting allocation rule is monotone and can be extended to a DSIC mechanism.[2]

2 Algorithmic Mechanism Design

2.1 The Holy Grail: DSIC For Free

Algorithmic mechanism design is one of the initial and most well-studied branches of algorithmic game theory, and we won't have time to do it justice. The dominant paradigm in algorithmic mechanism design is to relax the second constraint (optimal surplus) as little as possible, subject to the first (DSIC) and third (polynomial-time) constraints. For single-parameter environments, Myerson's Lemma implies that the following goal is equivalent: design a polynomial-time and monotone allocation rule that comes as close as possible to maximizing the social surplus.

[2]Be warned, though, that the payments also need to be computed, which generally requires solving n more surplus-maximization problems (one per player). See also Exercise 16.

One of the reasons there has been so much progress in algorithmic mechanism design over the past 15 years is that, on a technical level, it bears a strong resemblance to the relatively mature field of *approximation algorithms*. The primary goal in approximation algorithms is to design algorithms for NP-hard problems that are as close to optimal as possible, subject to a polynomial-time constraint. Algorithmic mechanism design (for single-parameter problems) has *exactly* the same goal, except the algorithms must additionally obey a monotonicity constraint. Myerson's Lemma implies that algorithmic mechanism design boils down to algorithm design in an oddly restricted (via monotonicity) "computational model" — the entire game-theoretic aspect of the design goal is neatly compiled into a relatively intuitive extra constraint on the allocation rule.

It should be clear what the "holy grail" in algorithmic mechanism design is: for as many NP-hard problems of interest as possible, to match the best-known approximation guarantee for (not necessarily monotone) approximate surplus maximization algorithms — or even the best-possible approximation guarantee, subject to $P \neq NP$. That is, we would like the DSIC/monotone constraint to cause no additional surplus loss, beyond the loss we already have to suffer due to the polynomial-time constraint. Recall that we've been spoiled so far: with *exact* surplus-maximization, the DSIC/monotone constraint is satisfied "for free", and exact surplus-maximization with unknown data reduces to exact surplus-maximization with known data. Does an analogous reduction hold for *approximate* surplus maximization?

2.2 Knapsack Auctions (Reprise)

To explore the question above in a concrete setting, let's return to knapsack auctions. There are a number of heuristics for the knapsack problem that have good worst-case performance guarantees. For example, consider the following allocation rule $\mathbf{x}^G$, 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. We assume, without loss of generality, that $w_i \leq W$ for every i (it is harmless to delete bidders i with $w_i > W$).

(1) Sort and re-index the bidders so that[3]

$$\frac{b_1}{w_1} \geq \frac{b_2}{w_2} \geq \cdots \geq \frac{b_n}{w_n}.$$

(2) Pick winners in this order until one doesn't fit, and then halt.[4]

(3) Return either the step-2 solution, or the highest bidder, whichever creates more surplus.[5]

[3] What makes a bidder attractive is a high bid and a small size. We trade these off by ordering bidders by "bang-per-buck" — the value contributed per unit of capacity used.

[4] One can also continue to follow the sorted order, packing any further bidders that happen to fit — this will only do better.

[5] The reason for this step is that the solution in step 2 might be highly suboptimal if there a very valuable and very large bidder. One can also sort the bidders in nondecreasing bid order and pack them greedily — this will only do better.

The above greedy algorithm is a $\frac{1}{2}$-approximation algorithm for the Knapsack problem, which gives the following guarantee.

Theorem 2.1 *Assuming truthful bids, the surplus of the greedy allocation rule is at least 50% of the maximum-posible surplus.*

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 relax the problem so that a bidder can be chosen *fractionally*, with its value pro-rated accordingly. For example, if 70% of a bidder with value 10 is chosen, then it contributes 7 to the surplus. 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, as needed). A straightforward exchange argument proves that this algorithm maximizes the surplus over all feasible solutions to the fractional knapsack problem.

Suppose in the optimal fractional solution, the first k bidders in the sorted order win and the $(k+1)$th bidder fractionally wins. The surplus achieved by steps (1) and (2) in the greedy allocation rule is exactly the total value of the first k bidders. The surplus achieved in step (3) in the greedy allocation rule is at least the total value of the $(k+1)$th bidder. The better of these two solutions is at least half of the surplus of the optimal fractional solution, which is at least the surplus of an optimal (non-fractional) solution to the original problem. ■

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.

We know that surplus-maximization yields a monotone allocation rule; what about *approximate* surplus-maximization? At least for the greedy allocation rule above, we still have monotonicity (Exercise 18).

You may have been lulled into complacency, thinking that every reasonable allocation rule is monotone. The only non-monotone rule we've seen in the "second-highest bidder wins" rule for single-item auctions, which we don't care about anyways. *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)". 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 (see the Problems for details). This is characteristic of work in algorithmic mechanism design: consider an NP-hard optimization problem, check if the state-of-the-art approximation algorithm directly leads to a DSIC mechanism and, if not, tweak it or design a new approximation algorithm that does, hopefully without degrading the approximation guarantee.

2.3 Black-Box Reductions

Algorithmic mechanism design has been extremely successful for the single-parameter problems we've been discussing so far. The state-of-the-art approximation algorithms for such problems are generally either monotone or can redesigned to be monotone, like in the case of knapsack auctions mentioned above and in the problems. This success has been so widespread as to suggest the question:

> Is there a natural single-parameter problem for which the best approximation guarantee achievable by a polynomial-time algorithm is strictly better than the best approximation guarantee achievable by a polynomial-time *and monotone* algorithm?

Of course, a negative answer would be especially exciting — it would imply that, as with exact surplus-maximization, the monotonicity/DSIC constraint can always be added "for free". One way of proving such a sweeping positive result would be via a "black-box reduction": a generic way of taking a possibly non-monotone polynomial-time algorithm and transmuting it into a monotone polynomial-time algorithm without degrading the approximation guarantee. Such a reduction would be very interesting even if the approximation factor suffered by a constant factor.

Recent work by Chawla et al. [1] shows that there is no fully general black-box reduction of the above type for single-parameter environments. There might well be large and important subclasses of such environments, though, for which a black-box reduction exists. For example, does such a reduction apply to all *downward-closed* environments where, like in all of our examples so far, giving a bidder less stuff cannot render an outcome infeasible?[6]

3 The Revelation Principle

3.1 The DSIC Condition, 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. 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 can also be useful in practice.

Can non-DSIC mechanisms accomplish things that DSIC mechanisms cannot? To answer this question, let's tease apart two separate assumptions that are conflated in our DSIC definition:

(1) Every participant in the mechanism has a dominant strategy, no matter what its private valuation is.

[6] If the DSIC constraint is weakened to an implementation in Bayes-Nash equilibrium, then there are quite general black-box reductions. We'll discuss these in more advanced lectures.

(2) This dominant strategy is *direct revelation*, where the participant truthfully reports all of its 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 its value.

3.2 Beyond Dominant-Strategy Equilibria

Suppose we relax condition (1). The drawback is that we then need stronger assumptions to predict the behavior of participants and the mechanism's outcome; for example, we might consider a Bayes-Nash equilibrium with respect to a common prior (see Problem 6 on first-price auctions) or a Nash equilibrium in a full-information model (see Problem 3 on the GSP sponsored search auction). But 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. We'll do this in more advanced lectures. A very rough rule of thumb is that, for sufficiently simple problems like those in our introductory lectures, DSIC mechanisms can do anything non-DSIC mechanisms can. In more complex problems, like some discussed in the advanced lectures, weakening the DSIC constraint (e.g., to implementation in Bayes-Nash equilibrium) often allows you accomplish things that are provably impossible for DSIC mechanisms (assuming participants figure out and coordinate on the desired equilibrium). 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 will depend on the details of the application.

3.3 The Revelation Principle and the Irrelevance of Truthfulness

The Revelation Principle states that, given requirement (1) in Section 3.1, there is no need to relax requirement (2): it comes "for free."

Theorem 3.1 (Revelation Principle) *For every mechanism M in which every participant has a dominant strategy (no matter what its private information), there is an equivalent direct-revelation DSIC mechanism M'.*

Proof: The proof uses a simulation argument; see Figure 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.

Construct the following mechanism M', to which participants delegate the responsibility of playing the appropriate dominant strategy. Precisely, (direct-revelation) mechanism M' accepts sealed bids $b_1, \ldots, b_n$ from the players. It submits the bids $s_1(b_1), \ldots, s_n(b_n)$ to the mechanism M, and chooses the same outcome (e.g., winners of an auction and selling prices) that M does.

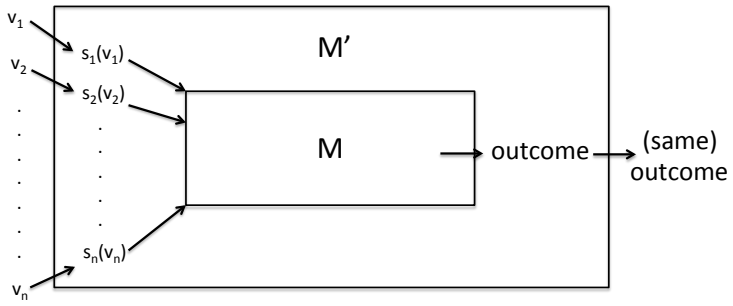


Figure 2: Proof of the Revelation Principle. Construction of the direct-revelation mechanism M', given a mechanism M with dominant strategies.

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 3.1 is that, at least in principle, if you design a mechanism to have dominant strategies, then you might as well design for direct revelation (in auctions, truthful bidding) to be a dominant strategy.

Many equilibrium concepts other than dominant-strategy equilibria, such as Bayes-Nash equilibria, have their own Revelation Principle. Such principles state that, given the choice of incentive constraints, direct revelation is without loss of generality. Thus, *truthfulness per se is not important*; what makes mechanism design hard is the requirement that a desired outcome (without loss of generality, truthful reporting) in an equilibrium of some type. Varying the choice of equilibrium concept can lead to quite different mechanism design theories, with stronger equilibrium concepts (like dominant-strategy equilibria) requiring weaker behavioral assumptions but with narrower reach than weaker equilibrium concepts (like Bayes-Nash equilibria).

References

[1] S. Chawla, N. Immorlica, and B. Lucier. On the limits of black-box reductions in mechanism design. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 435–448, 2012.

CS364A: Algorithmic Game Theory
Lecture #5: Revenue-Maximizing Auctions*

Tim Roughgarden†

October 7, 2013

1 The Challenge of Revenue Maximization

1.1 Welfare-Maximization, Revisited

Thus far, we've focused on the design of mechanisms that maximize, exactly or approximately, the *welfare* objective

$$\sum_{i=1}^{n} v_i x_i \tag{1}$$

over all feasible outcomes $(x_1, \ldots, x_n)$ in some set X. Revenue is generated in welfare-maximizing auctions, but only as a side effect, a necessary evil to incentivize participants to report their private information. This lecture begins our discussion of auctions that are explicitly designed to raise as much revenue as possible.

We started with the welfare objective for several reasons. One is that it's a fundamental objective function, relevant to many real-world scenarios. For instance, in government auctions (e.g., to sell wireless spectrum), the primary objective is welfare maximization — revenue is also important but is usually not the first-order objective. Also, in competitive markets, it is often thought that a seller should focus on welfare-maximization, since otherwise someone else will (potentially stealing their customers).

The second reason we started with welfare-maximization is pedagogical: welfare is special. In every single-parameter environment (and even more generally, see Lecture #7), there is a DSIC mechanism for maximizing welfare *ex post* — as well as if the designer knew all of the private information (the v_i's) in advance. In this sense, one can satisfy the DSIC constraint "for free." This is an amazingly strong performance guarantee, and it cannot generally be achieved for other objective functions.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

1.2 One Bidder and One Item

The following trivial example is illuminating. Suppose there is one item and only one bidder, with a private valuation v. With only one bidder, the space of direct-revelation DSIC auctions is small: they are precisely the "posted prices", or take-it-or-leave-it offers.[1] If the seller posts a price of r, then its revenue is either r (if $v \geq r$) or 0 (if $v < r$).

Maximizing the welfare in this setting is trivial: just set $r = 0$, so that you always give the item to the bidder for free. Note that this optimal posted price if *independent of* v.

Suppose we wanted to maximize revenue. How should we set r? Ex post (i.e., telepathically knowing v), we would set $r = v$. But with v private, what should we do? It's not obvious how to reason about this question.

The fundamental issue is that, for the revenue objective, different auctions do better on different inputs. With a single item and bidder, a posted price of 20 will do very well on inputs where v is 20 or a little larger, and terribly on smaller inputs (for which smaller posted prices will do better). Such trade-offs are familiar to students of algorithms. For example, different sorting algorithms (e.g., InsertionSort vs. QuickSort) run faster on different inputs. Different heuristics for the Traveling Salesman Problem (e.g., local search vs. linear programming) have smaller solution error on different inputs. Welfare-maximization, where there is an input-independent optimal DSIC mechanism, is special indeed.

1.3 Bayesian Analysis

Comparing different auctions for revenue maximization requires a model to reason about trade-offs across different inputs. Today we introduce the most classical and well-studied model for doing this: *average-case* or *Bayesian* analysis. Our model comprises the following ingredients:

- A single-parameter environment (see Lecture #3).
- The private valuation v_i of participant i is assumed to be drawn from a distribution F_i with density function f_i with support contained in $[0, v_{\max}]$.[2] We assume that the distributions $F_1, \ldots, F_n$ are independent (but not necessarily identical). In practice, these distributions are typically derived from data, such as bids in past auctions.
- The distributions $F_1, \ldots, F_n$ are known in advance to the mechanism designer. The realizations $v_1, \ldots, v_n$ of bidders' valuations are private, as usual. Since we focus on DSIC auctions, where bidders have dominant strategies, the bidders do not need to know the distributions $F_1, \ldots, F_n$.[3]

[1] Precisely, these are the deterministic such auctions. One can also randomize over posted prices, but the point of the example remains the same.

[2] Recall $F_i(z)$ denotes the probability that a random variable drawn from F_i has value at most z.

[3] The main results in today's lecture apply more generally to "Bayes-Nash incentive-compatible" auctions; in this case, the bidders must also know the distributions $F_1, \ldots, F_n$.

In a Bayesian environment, it is clear how to define the "optimal" auction — it is the auction that, among all DSIC auctions, has the highest expected revenue, where the expectation is with respect to the given distribution $F_1 \times F_2 \times \cdots \times F_n$ over valuation profiles $\mathbf{v}$ (assuming truthful bidding).

1.4 One Bidder and One Item, Revisited

With our Bayesian model, single-bidder single-item auctions are now easy to reason about. The expected revenue of a posted price r is simply

$$\underbrace{r}_{\text{revenue of a sale}} \cdot \underbrace{(1 - F(r))}_{\text{probability of a sale}}.$$

Given a distribution F, it is usually a simple matter to solve for the best r. The optimal posted price is called the *monopoly price* of the distribution F. Since DSIC mechanisms are posted prices (and randomizations thereof), posting the monopoly price is the revenue-maximizing auction. For instance, if F is the uniform distribution on $[0, 1]$ (i.e., $F(x) = x$ on $[0, 1]$), then the monopoly price is $\frac{1}{2}$, achieving an expected revenue of $\frac{1}{4}$.

The plot thickens even with two bidders, where the space of DSIC auctions is larger. For example, consider a single-item auction with two bidders with valuations drawn i.i.d. from the uniform distribution on $[0, 1]$. We could of course run the Vickrey auction; its revenue is the expected value of the smaller bid, which is $\frac{1}{3}$ (exercise).

We could also supplement the Vickrey auction with a *reserve price*, analogous to the "opening bid" in an eBay auction. In a Vickrey auction with reserve r, the allocation rule awards the item to the highest bidder, unless all bids are less than r, in which case no one gets the item. The corresponding payment rule charges the winner (if any) the second-highest bid or r, whichever is larger. From a revenue vantagepoint, adding a reserve price r is both good and bad: you lose revenue when all bids are less than r, but you gain revenue when exactly one bid is above r (since the selling price is higher). In our case, adding a reserve price of $\frac{1}{2}$ turns out to be a net gain, raising the expected revenue from $\frac{1}{3}$ to $\frac{5}{12}$ (exercise). But can we do better? Perhaps using a different reserve price, or perhaps usually a totally different auction format? While the rich space of DSIC auctions makes this an intimidating question, the rest of this lecture provides a complete solution, originally given by Myerson [2].

2 Expected Revenue Equals Expected Virtual Welfare

Our goal is to characterize the optimal (i.e., expected revenue-maximizing) DSIC auction for every single-parameter environment and distributions $F_1, \ldots, F_n$.[4] We begin with a preliminary observation.

[4] We won't prove it today, but the auctions we identify are optimal in a much stronger sense; see the discussion in Section 3.2.

Step 0: By the Revelation Principle from last lecture, every DSIC auction is equivalent to — and hence has the same expected revenue as — a direct-revelation DSIC mechanism $(\mathbf{x}, \mathbf{p})$. We can therefore consider only direct-revelation mechanisms from here on. We correspondingly assume truthful bids (i.e., $\mathbf{b} = \mathbf{v}$) for the rest of the lecture.

The expected revenue of an auction $(\mathbf{x}, \mathbf{p})$ is

$$\mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} \mathbf{p}(\mathbf{v})\right], \tag{2}$$

where the expectation is with respect to the distribution $F_1 \times \cdots \times F_n$ over bidders' valuations. It is not clear how to directly maximize the expression (2) over the space of DSIC mechanisms. In this section, we derive a *second* formula for the expected revenue of an auction. This second formula only references the allocation rule of a mechanism, and not its payment rule, and for this reason has a form that is far easier to maximize.

As a starting point, recall Myerson's payment formula from Lecture #3:

$$p_i(b_i, \mathbf{b}_{-i}) = \int_0^{b_i} z \cdot x_i'(z, \mathbf{b}_{-i}) dz. \tag{3}$$

We derived this equation assuming that the allocation function $x_i(z, \mathbf{b}_{-i})$ is differentiable. By standard advanced calculus, the same formula holds more generally for an arbitrary monotone function $x_i(z, \mathbf{b}_{-i})$, including piecewise constant functions, for a suitable interpretation of the derivative $x_i'(z, \mathbf{b}_{-i})$ and the corresponding integral. Similarly, all of the following proof steps, which make use of calculus maneuvers like integration by parts, can be made fully rigorous for arbitrary bounded monotone functions without significant difficulty. We leave the details to the interested reader.[5]

Equation (3) states that payments are fully dictated by the allocation rule. Thus, at least in principle, we can express the expected revenue of an auction purely in terms of its allocation rule, with no explicit reference to its payment rule. Will the resulting revenue formula will be easier to maximize than the original one? It's hard to know without actually doing it, so let's do it.

Step 1: Fix i and $\mathbf{v}_{-i}$; recall that $\mathbf{v}_{-i}$ is a random variable (as is v_i), and we'll integrate out over it later.

By Myerson's payment formula (3), we can write the expected payment by bidder i for a given value of $\mathbf{v}_{-i}$ as

$$\mathbf{E}_{v_i \sim F_i}[p_i(\mathbf{v})] = \int_0^{v_{\max}} p_i(\mathbf{v}) f_i(v_i) dv_i = \int_0^{v_{\max}} \left[\int_0^{v_i} z \cdot x_i'(z, \mathbf{v}_{-i}) dz\right] f_i(v_i) dv_i.$$

Note that in the first equality we're exploiting the independence of bidders' valuations — the fixed value of $\mathbf{v}_{-i}$ has no bearing on the distribution F_i from which v_i is drawn.

[5] For example, every bounded monotone function is integrable, and is differentiable except on a set of measure zero.

This step is exactly what we knew was possible in principle — rewriting the payment in terms of the allocation rule. For this to be useful, we need some simplifications.

Step 2: Whenever you have a double integral (or double sum) that you don't know how to interpret, it's worth reversing the integration order. Here, reversing the order of integration leads to a nice simplification, suggesting we're on the right track:

$$\begin{aligned}\int_0^{v_{\max}} \left[\int_0^{v_i} z \cdot x_i'(z, \mathbf{v}_{-i}) dz\right] f_i(v_i) dv_i &= \int_0^{v_{\max}} \left[\int_z^{v_{\max}} f_i(v_i) dv_i\right] z \cdot x_i'(z, \mathbf{v}_{-i}) dz \\ &= \int_0^{v_{\max}} (1 - F_i(z)) \cdot z \cdot x_i'(z, \mathbf{v}_{-i}) dz.\end{aligned}$$

Step 3: Integration by parts is also worth trying when attempting to massage an integral into a more interpretable form, especially if there's an obvious derivative hiding in the integrand. Here, we again get some encouraging simplifications:

$$\begin{aligned}&\int_0^{v_{\max}} \underbrace{(1 - F_i(z)) \cdot z}_{f} \cdot \underbrace{x_i'(z, \mathbf{v}_{-i})}_{g'} dz \\ &= \underbrace{(1 - F_i(z)) \cdot z \cdot x_i(z, \mathbf{v}_{-i})\big|_0^{v_{\max}}}_{=0-0} - \int_0^{v_{\max}} x_i(z, \mathbf{v}_{-i}) \cdot (1 - F_i(z) - z f_i(z)) dz \\ &= \int_0^{v_{\max}} \underbrace{\left(z - \frac{1 - F_i(z)}{f_i(z)}\right)}_{:=\varphi_i(z)} x_i(z, \mathbf{v}_{-i}) f_i(z) dz.\end{aligned}$$

Notice that we can interpret the final expression as an expected value, where z is drawn from the distribution F_i.

Step 4: To simplify and help interpret the expression above, we introduce some new notation. The *virtual valuation* $\varphi_i(v_i)$ of bidder i with valuation v_i drawn from F_i is

$$\varphi_i(v_i) = v_i - \tfrac{1 - F_i(v_i)}{f_i(v_i)}.$$

Note that the virtual valuation of a bidder depends on its own valuation and distribution, and not on those of the others.

For example, consider a bidder i with valuation drawn from the uniform distribution on $[0, 1]$. Then $F_i(z) = z$, $f_i(z) = 1$, and $\varphi_i(z) = z - \frac{1-z}{1} = 2z - 1$ in $[0, 1]$. Notice that a virtual valuation can be negative! See the exercises for more examples.

Steps 1–4 Summary: For every bidder i and valuations $\mathbf{v}_{-i}$,

$$\mathbf{E}_{v_i \sim F_i}[p_i(\mathbf{v})] = \mathbf{E}_{v_i \sim F_i}[\varphi_i(v_i) \cdot x_i(\mathbf{v})]. \tag{4}$$

Remark 2.1 Virtual valuations play a central role in the design of Bayesian optimal auctions. Is there any intuition for what they mean? One coarse way to interpret the formula

$$\varphi_i(v_i) = \underbrace{v_i}_{\text{what you'd like to charge } i} - \underbrace{\frac{1 - F_i(v_i)}{f_i(v_i)}}_{\text{"information rent" earned by bidder } i}$$

is to think of v_i as the maximum revenue obtainable from bidder i, and the second term as the inevitable revenue loss caused by not knowing v_i in advance (a.k.a. "information rent"). A second and more accurate interpretation of $\varphi_i(v_i)$ is as the slope of a "revenue curve" at v_i, where the revenue curve plots the expected revenue obtained from an agent with valuation drawn from F_i, as a function of the probability of a sale. The exercises elaborate on this second interpretation.

Step 5: Take the expectation, with respect to $\mathbf{v}_{-i}$, of both sides of (4) to obtain:

$$\mathbf{E}_{\mathbf{v}}[p_i(\mathbf{v})] = \mathbf{E}_{\mathbf{v}}[\varphi_i(v_i) \cdot x_i(\mathbf{v})]\,.$$

Step 6: Apply linearity of expectations (twice) to finish the derivation:

$$\mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} p_i(\mathbf{v})\right] = \sum_{i=1}^{n} \mathbf{E}_{\mathbf{v}}[p_i(\mathbf{v})] = \sum_{i=1}^{n} \mathbf{E}_{\mathbf{v}}[\varphi_i(v_i) \cdot x_i(\mathbf{v})] = \mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} \varphi_i(v_i) \cdot x_i(\mathbf{v})\right]. \tag{5}$$

The final term in (5) is our second formula for the expected revenue of an auction, and we should be pleased with its relative simplicity. Note that if we removed the φ_i's from the expression, we would be left with an old friend: the expected *welfare* of the auction. For this reason, we refer to $\sum_{i=1}^{n} \varphi_i(v_i) \cdot x_i(\mathbf{v})$ as the *virtual welfare* of an auction on the valuation profile $\mathbf{v}$. We have proved that, for every auction,

$$\text{EXPECTED REVENUE} = \text{EXPECTED VIRTUAL WELFARE}. \tag{6}$$

In particular, maximizing expected revenue over the space of DSIC auctions reduces to maximizing expected virtual welfare.

3 Bayesian Optimal Auctions

It is shocking that a formula as simple as (6) holds. It says that even though we only care about payments, we can focus on an optimization problem that concerns only the mechanism's allocation rule. This second form is far more operational, and we proceed to determine the auctions that maximize it.

3.1 Maximizing Expected Virtual Welfare

As a warm up, let's make two extra assumptions. First, consider a single-item auction. Second, assume that the bidders are i.i.d. That is, all F_i's are a common F, and thus all virtual valuation functions φ_i are the same.

How should we choose the allocation rule $\mathbf{x}$ to maximize the expected virtual welfare

$$\mathbf{E}_{\mathbf{v}\sim\mathbf{F}}\left[\sum_{i=1}^{n}\varphi_i(v_i)x_i(\mathbf{v})\right]? \tag{7}$$

We have the freedom of choosing $\mathbf{x}(\mathbf{v})$ for each input $\mathbf{v}$, and have no control over the input distribution $\mathbf{F}$ or the virtual values $\varphi_i(v_i)$. Thus, the obvious approach is to maximize pointwise: separately for each input $\mathbf{v}$, we choose $\mathbf{x}(\mathbf{v})$ to maximize the virtual welfare $\sum_{i=1}^{n}\varphi_i(v_i)x_i(\mathbf{v})$ obtained on the input $\mathbf{v}$ (subject to feasibility of $(x_1,\ldots,x_n)\in X$). We call this the *virtual welfare-maximizing allocation rule*.

In a single-item auction, where the feasibility constraint is $\sum_{i=1}^{n}x_i(\mathbf{v})\leq 1$ for each $\mathbf{v}$, the virtual welfare-maximizing rule just awards the item to the bidder with the highest virtual valuation. Well not quite: recall that virtual valuations can be negative (e.g., $\varphi_i(v_i)=2v_i-1$ when v_i is uniform between 0 and 1), and if every bidder has a negative virtual valuation then the virtual welfare is maximized by not awarding the item to anyone. (We already saw in the single-bidder example that maximizing revenue entails not selling the item in some cases.)

Choosing $\mathbf{x}(\mathbf{v})$ separately for each $\mathbf{v}$ to maximize $\sum_{i=1}^{n}\varphi_i(v_i)x_i(\mathbf{v})$ defines an allocation rule that maximizes the expected virtual welfare (7) over all allocation rules (monotone or not). The key question is: *is this virtual welfare-maximizing rule monotone?* If so, then it can be extended to a DSIC auction, and by (6) this auction has the maximum-possible expected revenue.

The answer to this key question depends on the valuation distribution F. If the corresponding virtual valuation function φ is increasing, then the virtual welfare-maximizing allocation rule is monotone.

Definition 3.1 A distribution F is *regular* if the corresponding virtual valuation function $v-\frac{1-F(v)}{f(v)}$ is strictly increasing.

For most applications, Definition 3.1 can be relaxed to allow nondecreasing virtual valuation functions.

We saw that the uniform distribution on $[0,1]$ has virtual valuation function $2v-1$ and hence is regular. So are other uniform distributions, exponential distributions, and lognormal distributions. Irregular distributions include many multi-modal distributions and distributions with sufficiently heavy tails. See the exercises for concrete examples.

Let's return to a single-item auction with i.i.d. bidders, under the additional assumption that the valuation distribution is regular. The virtual-welfare maximizing allocation rule, which allocates to the bidder with highest nonnegative virtual valuation (if any), is monotone and yields the optimal auction. Moreover, since all bidders share the same increasing virtual

valuation function, the bidder with the highest virtual valuation is also the bidder with the highest valuation. This allocation rule is thus equivalent to the Vickrey auction with a reserve price of $\varphi^{-1}(0)$. Thus, for i.i.d. bidders and a regular valuation distribution, eBay (with a suitable opening bid) is the optimal auction format! Given the richness of the DSIC auction design space, it is amazing that such a simple and practical auction pops out as the optimal one.

More generally, consider an arbitrary single-parameter environment and valuation distributions $F_1, \ldots, F_n$. The virtual welfare-maximizing allocation rule is now defined as that which, for each input $\mathbf{v}$, chooses the feasible allocation that maximizes the virtual welfare $\sum_{i=1}^{n} \varphi_i(v_i)x_i(\mathbf{v})$. If every distribution F_i is regular, then this allocation rule is monotone (see the exercises). Coupling it with the unique payment rule to meet the DSIC constraint, we obtain the optimal auction. In this sense, we have solved the Bayesian optimal auction problem for every single-parameter environment with regular valuation distributions.

3.2 Extensions

The theory developed in this lecture, which is due to Myerson [2], is even more general. First, it can be extended to accommodate valuation distributions that are not regular.[6] Since the virtual welfare-maximization allocation rule is not monotone in this case, one has to work harder and solve for the monotone allocation rule with the maximum expected virtual welfare. This can be done by "ironing" virtual valuation functions to make them monotone, while at the same time preserving the virtual welfare of the auctions that matter. See Hartline [1, Chapter 3] for a textbook treatment.

Second, while today's lecture restricted attention to DSIC auctions for simplicity, the (DSIC) optimal auctions we identified are optimal even amongst the much larger set of "Bayesian incentive compatible" mechanisms. For example, first-price auction formats cannot achieve more revenue (at equilibrium) than the best DSIC auction. Thus, for revenue-maximization in single-parameter problems, the DSIC constraint comes for free. This extension does not require significant new ideas beyond what we covered today, and we'll discuss it in CS364B.

References

[1] J. D. Hartline. Mechanism design and approximation. Book draft. October, 2013.

[2] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[6] Independence of the distributions, however, is crucial and cannot be relaxed without significantly changing the results.

CS364A: Algorithmic Game Theory
Lecture #6: Simple Near-Optimal Auctions*

Tim Roughgarden†

October 9, 2013

1 Optimal Auctions Can Be Complex

Last lecture we proved some of the most fundamental results in auction theory. To reiterate, for every DSIC auction $(\mathbf{x}, \mathbf{p})$ for a single-parameter environment with valuation distributions $F_1, \ldots, F_n$, the expected revenue equals the expected virtual welfare:

$$\mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} p_i(\mathbf{v})\right] = \mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} \varphi_i(v_i) \cdot x_i(\mathbf{v})\right]. \tag{1}$$

Define the virtual welfare-maximizing allocation rule as the one that sets

$$\mathbf{x}(\mathbf{v}) := \operatorname*{argmax}_{X} \sum_{i=1}^{n} \varphi_i(v_i) x_i(\mathbf{v})$$

for each input $\mathbf{v}$. If every F_i is regular, meaning that the corresponding virtual valuation function

$$\varphi_i(v_i) = v_i - \tfrac{1-F_i(v_i)}{f_i(v_i)}$$

is strictly increasing, then the virtual welfare-maximizing allocation rule is monotone and, after we define suitable payments, maximizes expected revenue over all DSIC auctions. This characterization of optimal auctions can be extended to irregular distributions, but this extension requires more work (see [3, Chapter 3]).

As a corollary of this general characterization, we noted that the optimal single-item auction with i.i.d. bidders and a regular distribution F is shockingly simple: it is simply the Vickrey auction, augmented with the reserve price $\varphi^{-1}(0)$. This is a true "killer application"

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

of auction theory — it gives crisp, conceptually clean, and practically useful guidance to auction design.

The plot thickens if we make the problem a bit more complex. Consider again a single-item auction, but relax the assumption that bidders' valuation distributions are identical; they are still independent and regular. The optimal auction can get weird, and it does not generally resemble any auctions used in practice (see the exercises). For example, someone other than the highest bidder might win. The payment made by the winner seems impossible to explain to someone who hasn't studied virtual valuations — compare this to the i.i.d. case, where the optimal auction is simply eBay with a smartly chosen opening bid. This weirdness is inevitable if you are 100% confident in your model (i.e., the F_i's) and you want every last cent of the maximum-possible expected revenue — there is no choice in which allocation or payment rule you can use.

Today we'll seek out auctions that are simpler, more practical, and more robust than the theoretically optimal auction. Since optimality requires complexity, we can only hope that our auctions are *approximately* optimal. This is the second time we've turned to approximation to escape a quandary posed by full optimality. In algorithm mechanism design (Lecture 4), we used approximation to recover computational tractability when the underlying welfare-maximization problem was NP-hard. Similarly, here we are rejecting the optimal auction because of its "complexity" and using approximation to recover relative simplicity. Unlike algorithmic mechanism design, where "low complexity" was precisely defined (as polynomial running time), here we leave terms like "simple", "practical", and "robust" largely undefined. Formulating definitions that capture these vague ideas and give useful guidance to the design of simple near-optimal auctions is an extremely important research question in the field.

The theory of Bayesian optimal auctions developed last lecture is part of the microeconomic canon, dating to 1981 [5]. By contrast, the research agendas outlined this lecture have been developed primarily over the past 5 years, mostly in the computer science literature. The classical theory from last lecture is the foundation on which this more recent work rests.

2 The Prophet Inequality

This section covers a fun result from optimal stopping theory. The next section uses it to design a relatively simple and provably near-optimal single-item auction for non-i.i.d. bidders.

Consider the following game, with has n stages. In stage i, you are offered a nonnegative prize π_i, drawn from a distribution G_i. You are told the distributions $G_1, \ldots, G_n$ in advance, and these distributions are independent. You are told the realization π_i only at stage i. After seeing π_i, you can either accept the prize and end the game, or discard the prize and proceed to the next stage. The decision's difficulty stems from the trade-off between the risk of accepting a reasonable prize early and then missing out later on a great one, and the risk of having to settle for a lousy prize in one of the final stages.

The Prophet Inequality, due to Samuel-Cahn [7], offers a simple strategy that does almost

as well as a fully clairvoyant prophet.

Theorem 2.1 (Prophet Inequality) *For every sequence $G_1, \ldots, G_n$ of independent distributions, there is strategy that guarantees expected reward $\frac{1}{2}\mathbf{E}_\pi[\max_i \pi_i]$. In fact, there is such a* threshold strategy *t, which accepts prize i if and only if $\pi_i \geq t$.*

Proof: Let z^+ denote $\max\{z, 0\}$. Consider a threshold strategy with threshold t. It is difficult to compare directly the expected payoff of this strategy with the expected payoff of a prophet. So, the plan is to derive lower and upper bounds, respectively, on these two quantities that are easily to compare.

Let $q(t)$ denote the probability that the threshold strategy accepts no prize at all.[1] As t increases, the risk $q(t)$ increases but the average value of an accepted prize goes up.

What payoff does the t-threshold strategy obtain? With probability $q(t)$, zero, and with probability $1 - q(t)$, at least t. Let's improve our lower bound in the second case. If exactly one prize i satisfies $\pi_i \geq t$, then we should get "extra credit" of $\pi_i - t$ above and beyond our baseline payoff of t. If at least two prizes exceed the threshold, say i and j, then things are more complicated: our "extra credit" is either $v_i - t$ or $v_j - t$, depending on which of i, j belongs to the earlier stage. Let's be lazy and punt on this complication: when two or more prizes exceed the threshold, we'll only credit the baseline t to the strategy's payoff.

Formally, we have the following lower bound:

$$\begin{aligned}
&\mathbf{E}[\text{payoff of } t\text{-threshold strategy}] \\
&\geq (1 - q(t)) \cdot t + \sum_{i=1}^{n} \mathbf{E}[\pi_i - t \,|\, \pi_i \geq t, \pi_j < t \,\forall j \neq i] \,\mathbf{Pr}[\pi_i \geq t] \cdot \mathbf{Pr}[\pi_j < t \,\forall j \neq i] \qquad (2) \\
&= (1 - q(t)) \cdot t + \sum_{i=1}^{n} \underbrace{\mathbf{E}[\pi_i - t \,|\, \pi_i \geq t] \,\mathbf{Pr}[\pi_i \geq t]}_{=\mathbf{E}[(\pi_i - t)^+]} \cdot \underbrace{\mathbf{Pr}[\pi_j < t \,\forall j \neq i]}_{\geq q(t)} \qquad (3) \\
&\geq (1 - q(t)) \cdot t + q(t) \sum_{i=1}^{n} \mathbf{E}\left[(\pi_i - t)^+\right], \qquad (4)
\end{aligned}$$

where we use the independence of the G_i's in (2) to factor the two probability terms and in (3) to drop the conditioning on the event that $\pi_j < t$ for every $j \neq i$. In (4), we use that $q(t) = \mathbf{Pr}[\pi_j < t \,\forall j] \leq \mathbf{Pr}[\pi_j < t \,\forall j \neq i]$.

Now we produce an upper bound on the prophet's expected payoff $\mathbf{E}[\max_i \pi_i]$ that is easy to compare to (4). The initial expression doesn't reference the strategy's threshold t, so we add and subtract it to derive

$$\begin{aligned}
\mathbf{E}\Big[\max_i \pi_i\Big] &= \mathbf{E}\Big[t + \max_i (\pi_i - t)\Big] \\
&\leq t + \mathbf{E}\Big[\max_i (\pi_i - t)^+\Big] \\
&\leq t + \sum_{i=1}^{n} \mathbf{E}\left[(\pi_i - t)^+\right]. \qquad (5)
\end{aligned}$$

[1] Note that discarding the final stage's prize is clearly suboptimal!

Comparing (4) and (5), we can set t so that $q(t) = \frac{1}{2}$ — i.e., there is a 50/50 chance of accepting a prize — to complete the proof.[2] ■

Our proof of Theorem 2.1 shows a stronger statement that is useful in the next section. Our lower bound (4) on the revenue of the t-threshold strategy only credits t units of value when at least two prizes exceed the threshold t; only realizations in which exactly one prize exceeds the threshold contribute to the second, "extra credit" term in (4). This means that the guarantee of $\frac{1}{2}\mathbf{E}[\max_i \pi_i]$ applies even if, whenever there are multiple prizes above the threshold, the strategy somehow picks the worst (i.e., smallest) of these.

3 Simple Single-Item Auctions

We now return to our motivating example of a single-item auction with n bidders with valuations drawn from (not necessarily identical) regular distributions $F_1, \ldots, F_n$. We use the Prophet Inequality to design a relatively simple, near-optimal auction.

The key idea is to regard the virtual valuation $\varphi_i(v_i)^+$ of a bidder, if nonnegative, as the ith prize. (G_i is then the corresponding distribution induced by F_i; since the F_i's are independent, so are the G_i's.) To see an initial connection to the Prophet Inequality, note that the expected revenue of the optimal auction is $\mathbf{E}_{\mathbf{v}}[\sum_i \varphi_i(v_i)x_i(\mathbf{v})] = \mathbf{E}_{\mathbf{v}}[\max_i \varphi_i(v_i)^+]$, precisely the expected value obtained by a prophet with prizes $\varphi_1(v_1)^+, \ldots, \varphi_n(v_n)^+$.

Now can consider any allocation rule that has the following form:

(1) Choose t such that $\mathbf{Pr}[\max_i \varphi_i(v_i)^+ \geq t] = \frac{1}{2}$.

(2) Give the item to a bidder i with $\varphi_i(v_i) \geq t$, if any, breaking ties among multiple candidate winners arbitrarily (subject to monotonicity).

The strong form of the Prophet Inequality immediately implies that every auction with an allocation rule of the above type satisfies

$$\mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{n} \varphi_i(v_i)^+ x_i(\mathbf{v})\right] \geq \frac{1}{2}\mathbf{E}_{\mathbf{v}}\left[\max_i \varphi_i(v_i)^+\right].$$

For example, here is a specific such allocation rule:

(1) Set a reserve price $r_i = {\varphi_i}^{-1}(t)$ for each bidder i, with t defined as above.

(2) Give the item to the highest bidder that meets its reserve (if any).

This auction first filters bidders using reserve prices, and then simply awards the item to the highest bidder remaining. This auction is qualitatively simpler than the optimal auction in two senses. First, the corresponding payment of the winning bidder is just the maximum of

[2] If there is no such t because of point masses in the G_i's, then a minor extension of the argument yields the same result (see Problem Set #2).

its reserve price and the highest bid by another bidder that meets its reserve price — thus, virtual valuation functions are only used to set reserve prices, and only the inverse virtual valuation of 0 matters. Second, the highest bidder wins, as long as it clears its reserve price.

This "simple" auction is more plausible to implement than an optimal auction, but an issue remains: the reserve prices are different for different bidders. Some real-world auctions use such non-anonymous reserve prices — in sponsored search auctions, "higher-quality" advertisers generally face lower reserve prices than lower-quality advertisers — but they are rare. On eBay, for example, you only get to set one opening bid, even if you know (from bidding histories, say) that the bidders are not i.i.d.

An interesting open research question is to understand how well the Vickrey auction with an anonymous reserve price (i.e., eBay) can approximate the optimal expected revenue in a single-item auction when bidders valuations are drawn from non-i.i.d. regular distributions. Partial results are known: there is such an auction that recovers at least 25% of the optimal revenue, and no such auction always recovers more than 50% of the optimal revenue [4].

More generally, designing simple auctions that provably approximate the optimal revenue has been a hot research topic for the past 5 years or so; see [3, Chapter 4] for a survey.

4 Prior-Independent Auctions and the Bulow-Klemperer Theorem

This section explores a different critique of the optimal auction approach developed last lecture: the valuation distributions $F_1, \ldots, F_n$ were assumed to be known to the seller in advance. In some applications, where there is lots of data and bidders' preferences are not changing too rapidly, this is a reasonable assumption. But what if the seller does not know, or is not confident about, the valuation distributions? This is a relevant issue in "thin markets" where there is not much data, including keyword auctions for rarely used (but potentially valuable) search queries.

Removing advance knowledge of the distributions might seem to banish us to our single-bidder single-item quandary (Lecture 5) that motivated the Bayesian approach. The difference is that we will continue to assume that bidders' valuations are drawn from distributions; it's just that these distribution are unknown a priori. That is, we now use distributions in the *analysis* of auctions, but not in their *design*. The goal is to design an auction, whose description is independent of the underlying distributions, that performs almost as well as if the distributions were known in advance. This research agenda of designing good *prior-independent* auctions was articulated by Dhangwatnotai et al. [2] and has been an active topic over the past three years; see Hartline [3, Chapter 5] for a survey of the latest developments.

Today, we'll cover a beautiful result from classical auction theory which is also an important precursor to the design of prior-independent auctions. The expected revenue of a Vickrey auction can obviously only be less than that of an optimal auction; yet the following result, due to Bulow and Klemperer [1], shows that this inequality reverses when the Vickrey auction's environment is made slightly more competitive.

Theorem 4.1 (Bulow-Klemperer Theorem [1]) *Let F be a regular distribution and n a positive integer. Then:*

$$\mathbf{E}_{v_1,\ldots,v_{n+1}\sim F}[\textit{Rev(VA) } (n+1 \textit{ bidders})] \geq \mathbf{E}_{v_1,\ldots,v_n\sim F}[\textit{Rev}(OPT_F) \textit{ } (n \textit{ bidders})], \qquad (6)$$

where VA and OPT_F denote the Vickrey auction and the optimal auction for F, respectively.[3]

Notice that the auction in the left-hand side of (6) — the Vickrey auction with no reserve — is "prior-independent," meaning its description is independent of the underlying distribution F. The auction in the right-hand side of (6) depends on the underlying distribution F through its reserve price. In this sense, a single auction (the Vickrey auction) is simultaneously competitive with an infinite number of different optimal auctions, across all possible single-item environments with i.i.d. regular bidder valuations. The guarantee in Theorem 4.1 also implies that, in every such environment with $n \geq 2$ bidders, the expected revenue of the Vickrey auction is at least $\frac{n-1}{n}$ times that of an optimal auction (for the same number of bidders); see the Exercises.

The usual interpretation of the Bulow-Klemperer theorem, which also has anecdotal support in practice, is that extra competition is more important than getting the auction format just right. That is, invest your resources into getting more serious participants, rather than sharpening your knowledge of their preferences (of course, do both if you can!). See the Problems for more extensions and variations of the Bulow-Klemperer theorem.

Proof of Theorem 4.1: The two sides of (6) are tricky to compare directly, so for the analysis we define a fictitious auction $\mathcal{A}$ to facilitate the comparison. This auction operates in the environment with $n+1$ bidders, as follows:

(1) Simulate the optimal auction OPT_F on the first n bidders $1, 2, \ldots, n$.

(2) If the item was not awarded in step 1, then give the item to bidder $n+1$ for free.

We defined $\mathcal{A}$ to possess two properties useful for the analysis. First, its expected revenue (with $n+1$ bidders) is exactly that of OPT_F (with n bidders). Second, $\mathcal{A}$ always allocates the item.

We can finish the proof by arguing that, for i.i.d. regular bidder valuations, the Vickrey auction maximizes expected revenue over all auctions that are guaranteed to allocate the item. By the equivalence of expected revenue and expected virtual welfare, the optimal auction that always allocates the item awards the item to the bidder with the highest virtual valuation (even if this is negative). The Vickrey auction awards the item to the bidder with the highest valuation. Since bidders' valuations are i.i.d. draws from a regular distribution, all bidders share the same increasing virtual valuation function φ. Thus the bidder with the highest virtual valuation is always the bidder with the highest valuation. We conclude that the Vickrey auction (with $n+1$ bidders) has expected revenue at least that of every auction that always allocates the item, including $\mathcal{A}$, and therefore its expected revenue is at least that of OPT_F (with n bidders). ■

[3]That is, OPT_F is the Vickrey auction with the monopoly reserve price $\varphi^{-1}(0)$, where φ is the virtual valuation function of F.

5 Case Study: Reserve Prices in Yahoo! Keyword Auctions

So how does all this optimal auction theory get used, anyway? We next discuss a 2008 field experiment by Ostovsky and Schwarz [6], which explored whether or not the lessons of auction theory could be used to increase revenue for Yahoo! in its keyword search auctions.

Recall from Lecture #2 the standard model of keyword auctions. Which such auction maximizes the expected revenue, at least in theory? Assuming that bidders' valuations-per-click are drawn i.i.d. from a regular distribution F, it is simply to rank bidders by bid (from the best slot to the worst) after applying the monopoly reserve price $\varphi^{-1}(0)$ to all bidders, where φ is the virtual valuation function of F. See the exercises for details.

What had Yahoo! been doing, up to 2008? First, they were using relatively low reserve prices — initially \$.01, later \$.05, and \$.10 in 2008. Perhaps more naively, they were using the same reserve price of \$.10 across all keywords, even though some keywords surely warranted higher reserve prices than others (e.g., compare the searches "divorce lawyer" with "pizza"). How would Yahoo!'s revenue change if reserve prices were changed, independently for each keyword, to be theoretically optimal?

The field experiment described in [6] had two parts. First, a lognormal valuation distribution was posited for each of a half million keywords based on past bidding data.[4] This step is somewhat ad hoc but there is no evidence that the final conclusions depend on its details (such as the particular family of distributions used).

Next, theoretically optimal reserve prices were computed for each keyword, assuming valuations are drawn from the fitted distributions. As expected, the optimal reserve price varies a lot across keywords, but there are plenty of keywords with a theoretically optimal reserve price of \$.30 or \$.40. Thus, Yahoo!'s uniform reserve price was much too low, relative to the theoretical advice, on many keywords.

The obvious experiment is to try out the theoretically optimal (and generally higher) reserve prices to see how they do. Yahoo!'s top brass wanted to be a little more conservative, though, and set the new reserve prices to be the average of the old ones (\$.10) and the theoretically optimal ones.[5] And the change worked: auction revenues went up several per cent (of a very large number). The new reserve prices were especially effective in markets that are valuable but "thin," meaning not very competitive (less than 6 bidders). Better reserve prices were credited by Yahoo!'s president as the biggest reason for higher search revenue in Yahoo!'s third-quarter report in 2008.

[4] Since Yahoo!, like other search engines, uses a non-DSIC auction based on the GSP auction (see Problem 3), one cannot expect the bids to be truthful. In [6], valuations are reversed engineered from the bids under the assumption that bidders are playing the equilibrium that is outcome-equivalent to the dominant-strategy outcome of the DSIC auction (as in Problem 3).

[5] It turns out that, both in theory and empirically, this initial change accounts for most of the revenue increase. Increasing the reserve price further does not have much effect on revenue.

References

[1] J. Bulow and P. Klemperer. Auctions versus negotiations. *American Economic Review*, 86(1):180–194, 1996.

[2] P. Dhangwatnotai, T. Roughgarden, and Q. Yan. Revenue maximization with a single sample. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 129–138, 2010.

[3] J. D. Hartline. Mechanism design and approximation. Book draft. October, 2013.

[4] J. D. Hartline and T. Roughgarden. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, pages 225–234, 2009.

[5] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[6] M. Ostrovsky and M. Schwarz. Reserve prices in internet advertising auctions: A field experiment. Working paper, December 2009.

[7] E. Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *Annals of Probability*, 12(4):1213–1216, 1984.

CS364A: Algorithmic Game Theory
Lecture #7: Multi-Parameter Mechanism Design and the VCG Mechanism*

Tim Roughgarden†

October 14, 2013

1 General Mechanism Design Problems

Previous lectures only considered single-parameter mechanism design problems, where each participant has just one piece of private information, its valuation per unit of stuff. In many problems, a participant has different private valuations for different items. Once we are unsure about whether a participant prefers item A to item B, for example, we are in the realm of *multi-parameter* mechanism design.

Here are the ingredients of a general, multi-parameter mechanism design problem:

- n strategic participants, or "agents;"
- a finite set Ω of outcomes;
- each agent i has a private valuation $v_i(\omega)$ for each outcome $\omega \in \Omega$.

The outcome set Ω is abstract and could be very large. In a single-item auction, Ω has only $n+1$ elements, corresponding to the winner of the item (if any). In the standard single-parameter model of a single-item auction, we assume that the valuation of an agent is 0 in all of the n outcomes in which it doesn't win, leaving only one unknown parameter per agent. In the more general multi-parameter framework above, an agent can have a different valuation for each possible winner of the auction. This example is not without plausible application: in a bidding war over a hot startup, for example, agent i's highest valuation might be for acquiring the startup, but if it loses it prefers that the startup be bought by a company in a different market, rather than by a direct competitor.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

2 The VCG Mechanism

Our next result is a cornerstone of mechanism design theory.

Theorem 2.1 (The Vickrey-Clarke-Groves (VCG) Mechanism [6, 1, 3]) *In every general mechanism design environment, there is a DSIC welfare-maximizing mechanism.*

Recall the three properties of "awesome" auctions from Lecture 2. Theorem 2.1 asserts the first two properties but not the third (polynomial running time). We already know that, even in single-parameter environments, we can't always have the second and third properties (unless $P = NP$). As we'll see, the VCG mechanism is highly non-awesome in many important applications.

As always, designing a (direct-revelation) DSIC mechanism is tricky because the allocation and payment rules need to be coupled carefully.[1] We apply the same two-step approach that served us so well in single-parameter environments.

The first step is to assume, without justification, that agents truthfully reveal their private information, and then figure out which outcome to pick. Since Theorem 2.1 demands welfare-maximization, the only solution is to pick the welfare-maximizing outcome, using bids as proxies for the true (and unknown) valuations. That is, given bids $\mathbf{b}_1, \ldots, \mathbf{b}_n$, where each $\mathbf{b}_i$ is indexed by Ω, we define the allocation rule $\mathbf{x}$ by

$$\mathbf{x}(\mathbf{b}) = \operatorname*{argmax}_{\omega \in \Omega} \sum_{i=1}^{n} b_i(\omega). \tag{1}$$

The second step is to define a payment rule that, when coupled with the above allocation rule, yields a DSIC mechanism. Last time we arrived at this point, for single-parameter environments, we formulated and proved Myerson's Lemma, which is a general solution to this second step for all such environments. Recall that Myerson's Lemma asserts that allocation rule monotonicity is necessary and sufficient for implementability and, for monotone rules, it gives an explicit formula for the unique payment rule that meets the DSIC condition. Myerson's Lemma does not hold beyond single-parameter environments — with an agent submitting bids in more than one dimension, it's not even clear how to define "monotonicity" of an allocation rule.[2] Similarly, the "critical bid" characterization of DSIC payments (for 0-1 problems) does not have an obvious analog in multi-parameter problems.

The key idea is to make use of an alternative characterization of DSIC payments for the welfare-maximizing allocation rule (proved in the exercises), as the "externality" caused by an agent i — the welfare loss inflicted on the other $n-1$ agents by i's presence. For example,

[1] The proof of the Revelation Principle in Lecture 4 holds without change in multi-parameter environments, so restricting to direct-revelation mechanisms is without loss of generality.

[2] There is an analogous characterization of implementable multi-parameter allocation rules in terms of "cycle monotonicity." This is an elegant result, analogous to the fact that a network admits well-defined shortest paths if and only if it possesses no negative cycle. Cycle monotonicity is far more unwieldy than single-parameter monotonicity, however. Because it is so brutal to verify, cycle monotonicity is rarely used to argue implementability or to derive DSIC payment rules in concrete settings.

in a single-item auction, the winning bidder inflicts a welfare loss of the second-highest bid to the others (assuming truthful bids), and this is precisely the Vickrey auction's payment rule. This idea of "charging an agent its externality" makes perfect sense in general mechanism design environments, and it corresponds to the payment rule

$$p_i(\mathbf{b}) = \underbrace{\max_{\omega \in \Omega} \sum_{j \neq i} b_j(\omega)}_{\text{without } i} - \underbrace{\sum_{j \neq i} b_j(\omega^*)}_{\text{with } i}, \tag{2}$$

where $\omega^* = \mathbf{x}(\mathbf{b})$ is the outcome chosen in (1). Note that $p_i(\mathbf{b})$ is always nonnegative (exercise).

We claim that this mechanism $(\mathbf{x}, \mathbf{p})$, the *VCG mechanism*, is DSIC. (By definition, it maximizes welfare assuming truthful bids.) For the first time since the Vickrey auction, we'll prove the DSIC property from scratch (i.e., without use of Myerson's Lemma). Recall this means that for every agent i and every set $\mathbf{b}_{-i}$ of bids by the other agents, agent i maximizes its quasilinear utility $v_i(\mathbf{x}(\mathbf{b})) - p_i(\mathbf{b})$ by setting $\mathbf{b}_i = \mathbf{v}_i$.

Fix i and $\mathbf{b}_{-i}$. When the chosen outcome $\mathbf{x}(\mathbf{b})$ is ω^*, i's utility is

$$v_i(\omega^*) - p_i(\mathbf{b}) = \underbrace{\left[v_i(\omega^*) + \sum_{j \neq i} b_j(\omega^*) \right]}_{\text{(A)}} - \underbrace{\left[\max_{\omega \in \Omega} \sum_{j \neq i} b_j(\omega) \right]}_{\text{(B)}}. \tag{3}$$

Observe that the term (B) is a constant, independent of what $\mathbf{b}_i$ is. Thus, the problem of maximizing agent i's payoff reduces to the problem of maximizing the first term (A). As a thought experiment, let's suppose agent i has the power to choose the outcome ω^* directly, rather than merely influencing the chosen outcome indirectly via its choice of bid $\mathbf{b}_i$. Agent i would, of course, use this extra power to choose an outcome that maximizes the term (A). If agent i sets $\mathbf{b}_i = \mathbf{v}_i$, then term (1) that the mechanism maximizes becomes identical to the term (A) that the agent wants maximized. Thus, bidding truthfully results in the mechanism choosing an outcome that maximizes agent i's utility; no other bid could be better. This completes the proof of Theorem 2.1.

Here is an alternative interpretation of the payments in the VCG mechanism. Rewrite the expression in (2) as

$$p_i(\mathbf{b}) = \underbrace{b_i(\omega^*)}_{\text{bid}} - \underbrace{\left[\sum_{j=1}^{n} b_j(\omega^*) - \max_{\omega \in \Omega} \sum_{j \neq i} b_j(\omega) \right]}_{\text{rebate}}. \tag{4}$$

We can thus think of agent i's payment as its bid minus a "rebate," equal to the increase in welfare attributable to i's presence. For example, in the Vickrey auction, the highest bidder pays its bid b_1 minus a rebate of $b_1 - b_2$ (where b_2 is the second-highest bid), the increase in welfare that the bidder brings to the table.

We leave it as an exercise to observe that the discount in (4) is always nonnegative, implying that $p_i(\omega^*) \leq b_i(\omega^*)$ and hence truthtelling agents are guaranteed nonnegative utility.

The upshot of the VCG mechanism is that, in general multi-parameter environments, DSIC welfare-maximization is always possible in principle. While it can be infeasible to implement in practice, the VCG mechanism nevertheless serves as a useful benchmark for other, more practical approaches.

3 Combinatorial Auctions

Combinatorial auctions are important in practice. Already in the domain of government spectrum auctions, dozens of such auctions have raised hundreds of billions of dollars of revenue. They have also been used for other applications such as allocating take-off and landing slots at airports. Combinatorial auctions are also notoriously difficult, in both theory and practice. Theoretical work has identified many impossibility results for what can be done with reasonable communication and computation. Practice has provided examples of badly designed combinatorial auctions with serious consequences, such as the 1990 New Zealand spectrum auction that raised merely \$36 million, a far cry from consultants' estimates of \$250 million (see [5, Chapter 1] for details).

3.1 The Model

A combinatorial auction has n bidders — for example, Verizon, AT & T, and several regional providers. There is a set M of m items, which are *not* identical — for example, a license awarding the right to broadcast on a certain frequency in a given geographic area. The outcome set Ω corresponds to n-vectors $(S_1, \ldots, S_n)$, with S_i denoting the set of items allocated to bidder i (its "bundle"), and with no item allocated twice. There are $(n+1)^m$ different outcomes. Each bidder i has a private valuation $v_i(S)$ for each bundle $S \subseteq M$ of items it might get. Thus, each bidder has 2^m private parameters. One generally assumes that $v_i(\emptyset) = 0$ and that $v_i(S) \leq v_i(T)$ whenever $S \subseteq T$ (i.e., "free disposal"). We'll discuss additional assumptions on valuations later.

The welfare of an outcome $(S_1, \ldots, S_n)$ is $\sum_{i=1}^n v_i(S_i)$. In principle, the VCG mechanism provides a DSIC solution for maximizing the welfare. This can be useful when bidders' valuations are sufficiently structured, as with "unit-demand" bidders (see the exercises). In general, however, there are several major impediments to implementing the VCG mechanism.

3.2 Challenges

The first major challenge of combinatorial auctions is that of *preference elicitation*. Each bidder has $2^m - 1$ private parameters, roughly a thousand when $m = 10$ and a million when $m = 20$. No bidder in their right mind would want to write down (or even figure out) that many bids. No seller would want to listen to that many bids. This exponential number of

private parameters makes the VCG mechanism, and every other direct-revelation mechanism, a nonstarter for combinatorial auctions in practice. Note that this challenge never arises in single-parameter mechanism design, where each bidder only has to communicate one number.

The utter absurdity of direct-revelation combinatorial auctions motivates *indirect* mechanisms, which learn information about bidders' preferences only on a "need-to-know" basis. We have not yet discussed any such mechanisms. The canonical indirect auction is the ascending English auction. You're familiar with this auction format from the movies — an auctioneer keeps track of the current price and tentative winner, and the auction stops when only one interested bidder remains.[3] Each bidder has a dominant strategy, which is to stay in the auction as long as the current price is below its valuation (the player might win for positive utility) and to drop out once the current price reaches its valuation (after which winning can only lead to negative utility). Assuming all bidders play these strategies, the outcome of the English ascending auction is the same as that of the Vickrey (sealed-bid) auction. The Vickrey auction is what you get when you apply the Revelation Principle (Lecture 4) to the English auction.

Indirect auctions are unavoidable for all but the smallest combinatorial auctions. We'll discuss the auction formats used in practice for wireless spectrum auctions next lecture.[4] An important question is: can indirect mechanisms achieve, at least in principle, non-trivial welfare guarantees while eliciting only a small amount of information (say, polynomial in n and m) from the bidders? There is nice theoretical work on this question, and the answer, roughly, is "no" for complex valuations and "yes" for sufficiently simple valuations. We'll discuss this more in the next lecture, and at length in CS364B. In practice, one hopes that bidders' valuations are sufficiently simple and/or that auction performance will be much better than the worst-case bounds.

The second challenge in designing practice combinatorial auctions is familiar from our discussion of algorithmic mechanism design (Lecture 4). Even when the first challenge is not an issue — for example, when bidders are single-parameter and direct revelation is practical — welfare-maximization can be an intractable problem. We encountered this issue with Knapsack auctions and in a couple of the Problems. This challenge is fundamental, and cannot be avoided by choosing a clever auction format. In practice, one hopes that combinatorial auctions compute allocations that are reasonably close to welfare-maximizing. This is impossible to check directly, since bidders' valuations are unknown and the combinatorial

[3]There are a few variants. The movies, and auction houses like Christie's and Soethby's, use an "open outcry" auction in which bidders can drop out and return, and can make "jump bids" to aggressively raise the current price. When doing mathematical analysis, the "Japanese" variant is usually more convenient: the auction begins at some opening price, which is publicly displayed and increases at a steady rate. Each bidder either chooses "in" or "out," and once a bidder drops out it cannot return. The winner is the last bidder in, and the sale price is the price at which the second-to-last bidder dropped out.

[4]Indirect auctions can also be useful in single-parameter settings like single-item auctions. Empirical studies show that bidders are more likely to play their dominant strategy in an English auction than in a sealed-bid second-price auction, where some bidders inexplicably overbid [4]. Second, ascending auctions leak less valuation information to the auctioneer. In a Vickrey auction, the auctioneer learns the highest bid; in an English auction, the auctioneer only learns a lower bound on the highest bid (the second-highest bid).

auctions used in practice are not DSIC and offer some opportunities for strategizing. Nevertheless, there are various simple "sanity checks" that can be applied to an auction outcome to suggest good welfare maximization. For example, did bidders successfully acquire sensible packages (e.g., spectrum licenses that are adjacent geographically or in the spectrum)? Did similar items sell for similar prices?

The third challenge applies to the VCG mechanism — which turns out to be essentially the unique DSIC welfare-maximizing mechanism — even when the first two challenges are not relevant (e.g., a single-parameter problem small enough that welfare maximization can be done in a reasonable amount of time). Namely, the VCG mechanism can have bad revenue and incentive properties, despite being DSIC.

For instance, suppose there are two bidders and two items, A and B. The first bidder only wants both items, so $v_1(AB) = 1$ and is 0 otherwise. The second bidder only wants item A, so $v_2(AB) = v_2(A) = 1$ and is 0 otherwise. The revenue of the VCG mechanism is 1 in this example (exercise). But now suppose we add a third bidder who only wants item B, so $v_3(AB) = v_3(B) = 1$. The maximum welfare has jumped to 2, but the VCG revenue has dropped to 0 (exercise)! The fact that the VCG mechanism has zero revenue in seemingly competitive environments is a dealbreaker in practice. The revenue non-monotonicity in this example also implies numerous incentive problems, including vulnerability to collusion and false-name bids (see the exercises). None of these issues plague the single-item Vickrey auction.

The first challenge begets a fourth. Almost all combinatorial auctions used in practice are iterative, comprising a sequence of rounds; we'll discuss details next lecture. Iterative auctions offer new opportunities for strategic behavior. For example, Cramton and Schwatz [2] found that, in an early and relatively uncompetitive spectrum auction, bidders used the low-order digits of their bids to effectively send messages to other bidders. Let's consider license #378 in that auction, for spectrum use rights in Rochester, MN. USWest and McLeod were battling it out for this license, with each repeatedly outbidding the other. Apparently, USWest tired of this bidding war and switched to a retaliatory strategy — bidding on a number of licenses in other geographical areas on which McLeod was the standing high bidder, and on which USWest had shown no interest in previous rounds. McLeod ultimately won back all of these licenses, but had to pay a higher price due to USWest's bids. To make sure its message came through loud and clear, all of USWest's retaliatory bids were a multiple of 1000 *plus 378* — presumably warning McLeod to get the hell out of the market for Rochester, or else. This particular type of signalling can be largely eliminated by forcing all bids to be multiples of a suitably large number, but other opportunities for undesirable strategic behavior remain.

References

[1] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[2] P. Cramton and J. Schwartz. Collusive bidding: Lessons from the fcc spectrum auctions. *Journal of Regulatory Economics*, 17(3):229–252, 2000.

[3] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[4] R. M. Harstad. Dominant strategy adoption and bidders' experience with pricing rules. *Experimental Economics*, 3:261–280, 2000.

[5] P.. Milgrom. *Putting Auction Theory to Work*. Cambridge, 2004.

[6] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

CS364A: Algorithmic Game Theory
Lecture #8: Combinatorial and Wireless Spectrum Auctions*

Tim Roughgarden†

October 16, 2013

1 Selling Items Separately

Recall that a combinatorial auction has n bidders and m non-identical items, with bidder i having a private valuation $v_i(S)$ for every bundle $S \subseteq M$ of items. Asking each bidder to report 2^m bids is absurd unless m is very small. Thus, for the first time in the course, we have no choice but to design and analyze indirect mechanisms, and especially iterative mechanisms that query bidders for relevant valuation information on a "need-to-know" basis. This entails relaxing both the DSIC guarantee and full welfare maximization — we will miss these properties, but have no alternative.

What other mechanisms can we try? Given that we need to sell multiple items, and don't want to elicit valuations for every bundle, the simplest mechanisms to try are those that sell the items separately, using some type of single-item auction for each. We could certainly implement such an auction if desired — all we need is one bid per bidder per item, which is arguably the minimum imaginable.

We'll pin down the precise auction format shortly, but first we should ask a more basic question: could selling items separately conceivably work, even in principle? There is lots of beautiful and clarifying theory on this question, some of which we'll cover later. For now we summarize the main take-aways from this theory.

There is a fundamental dichotomy between combinatorial auctions in which items are *substitutes*, and those in which items are *complements* — with the former being far easier, in theory and in practice, than the latter. Roughly speaking, items are substitutes if you get diminishing returns from them — having one item only makes others less valuable. For two items A and B, for example, the substitutes condition means that $v(AB) \leq v(A) + v(B)$.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

In a spectrum auction context, two licenses for the same area with equal-sized frequency ranges are usually substitute items. Theory indicates that selling items separately has a chance to work well when items are (mostly) substitutes. For starters, welfare maximization is a computationally tractable problem when items are substitutes and the true valuations are known. In addition, the undesirable properties of the VCG mechanism pointed out last lecture and in the exercises evaporate when items are substitutes, generalizing the single-item case. But even though substitute items are the "easy" case, we'll see that it's easy to screw up when trying to sell them separately.

Item are complements if there are synergies between them, so that possessing one makes others more valuable. With two items A and B, this translates to the property $v(AB) > v(A) + v(B)$. Complements arise naturally in wireless spectrum auctions, as some bidders want a collection of licenses that are adjacent, either in their geographic areas or in their frequency ranges. With complements, welfare maximization (without incentive constraints) is already a very difficult problem; see also Problem 5. We cannot expect a simple auction format like separate single-item auctions to perform well in such environments.

The items in spectrum auctions, and most real-world combinatorial auctions, are a mixture of substitutes and complements. If the problem is "mostly substitutes," then separate single-item auctions might already perform well, if properly implemented. If not, then additional ideas are needed; see Section 3.

2 Simultaneous Ascending Auctions

There are numerous ways to organize separate single-item auction. Next we discuss two of the design decisions that seem to matter a lot in practice; see Cramton [2] and Milgrom [4, Chapter 1] for more details.

Rookie mistake #1: Hold the single-item auctions sequentially, one at a time.

To see why holding auctions sequentially is probably a bad idea, consider the especially easy case of identical items, where each bidder wants at most one. This problem can be solved easily via a single auction that allocates all of the items (e.g., by extending the Vickrey auction to this setting). Suppose instead we hold a sequence of single-item auctions. Concretely, consider two identical items, sold via back-to-back Vickrey auctions, and suppose you are a bidder with a very high valuation — you expect to win any auction that you participate in. What should you do? First, suppose everyone else bids straightforwardly, meaning that, if they haven't won an item yet, then they participate in the next auction and bid their true valuation. If you participate in the first auction, you would win and pay the second-highest valuation. If you skip it, the bidder with the second-highest valuation would win the first auction and disappear, leaving you to win the second auction at a price equal to the third-highest original valuation. Of course, now that we realize that it is not a dominant strategy for bidders to bid straightforwardly in a sequence of Vickrey auctions, we have to reason about how they might be strategizing. Summarizing, it's hard to bid intelligently in a sequence of Vickrey auctions because you have to guess the expected selling price of future

auctions, and this in turn makes the auctions' outcomes unpredictable, with the possibility of low welfare and revenue.

In March 2000, Switzerland auctioned off 3 blocks of spectrum via a sequence of Vickrey auctions. The first two auctions were for identical items, 28 MHz blocks, and sold for 121 million and 134 million Swiss francs, respectively. This is already more price variation than one would like for identical items. But the kicker was that in the third auction, where a larger 56 MHz block was being sold, the selling price was only 55 million! The bids were surely far from equilibrium, and both the welfare and revenue achieved by this auction are suspect.[1]

The discussion and history lessons above suggest holding single-item auctions for multiple items *simultaneously*, rather than sequentially. But there is still the question of the auction format for each single-item auction.

Rookie mistake #2: Use sealed-bid single-item auctions.

In 1990, the New Zealand government auctioned off essentially identical licenses for television broadcasting using simultaneous (sealed-bid) Vickrey auctions. It is again difficult for bidders to figure out how to bid in such an auction. Imagine, for example, that there are 10 licenses and you want one of them (but not more). How should you bid? One legitimate strategy is to pick one of the licenses (at random, say) and go for it. Another strategy is to bid less aggressively on multiple licenses, hoping that you get one at a bargain price, and that you don't win too many extra licenses that you don't want. The difficulty is trading off the risk of winning too many licenses with the risk of winning too few.

The difficulty of bidding and coordinating in a simultaneous sealed-bid auction makes the auction format vulnerable to outcomes with low welfare and revenue. For example, suppose there are three bidders and two identical items, and each bidder wants only one. The obvious extension of the Vickrey auction sells the two licenses to the bidders with the highest valuations, each at a price equal to the smallest valuation. In a simultaneous sealed-bid auction, if each bidder targets only one license, then one of the licenses is likely to have only one bidder and will thus be given away for free (or more generally, sold at the reserve price).

The revenue in the 1990 New Zealand auction was only $36 million, a paltry fraction of the projected $250 million. In contrast, most spectrum auctions over the past 20 years have met or exceedes projected revenues. On one license, the high bid was $100,000 while the second-highest bid (and selling price) was $6! On another, the high bid was $7 million and the second-highest was $5,000. To add insult to injury, the high bid were made available to the public, who could then see just how much money was left on the table! A later New Zealand auction kept the simultaneous sealed-bid format but switched to first-price auctions — this switch probably failed to prevent the miscoordination and consequent welfare and revenue losses that plagued the previous auction, but it did make these losses less evident to the public.

Simultaneous *ascending* auctions (SAAs) form the basis of most spectrum auctions run

[1] In addition to the questionable auction format, it didn't help matters that there were some strategic mergers of potential bidders before the auction, leading to less competition than expected.

over the last 20 years. We discuss the basic format first, and then some of the bells and whistles that have been added on over the years. Conceptually, SAAs are like a bunch of single-item English auctions being run in parallel in the same room, with one auctioneer per item. More precisely, each round, each bidder can place a new bid on any subset of items that it wants, subject to an *activity rule*. The activity rule forces all bidders to participate in the auction from the beginning and contribute to the price discovery discussed below. The details of an activity rule can be complex, but the gist is to require that the number of items that a bidder bids on only decreases over time as prices rise. Generally, the high bids and bidders are visible to all — even though this can encourage signaling and retaliatory bids (recall USWest vs. McLeod last lecture). The first round with no new bids ends the auction.

The main reason that SAAs work better than sequential or sealed-bid auctions is *price discovery*. As a bidder acquires better information about the likely selling prices of licenses, it can implement mid-course corrections — abandoning licenses for which competition is more fierce than anticipated, snapping up unexpected bargains, and rethinking which packages of licenses to assemble. The format typically resolves the miscoordination problems that plague simultaneous sealed-bid auctions. For instance, suppose there are two identical items and three bidders. Every round, some bidder will be losing both auctions. When it jumps back in, it makes sense to bid for the currently cheaper item, and this will keep the prices of the two items roughly the same.

Another bonus of the SAA format is that bidders only need to determine valuations on a need-to-know basis. We've been assuming that valuations are known to bidders at the beginning of the auction, but in practice determining the valuation for a bundle of items can be costly, involving research, expert advice, and so on. In sharp contrast to direct-revelation auctions, a bidder can often navigate an SAA with only coarse estimates for most valuations and precise estimates for the bundles that matter.

Generally, SAAs are believed to perform well, meaning they achieve good welfare and revenue. This assertion is not easy to test after an auction, since valuations remain unknown and bids are incomplete and potentially non-truthful. However, there are a number of "sanity checks" that suggest good auction performance. First, there should be little or no resale of items after the auction, and any reselling should take place at a price comparable to the auction's selling price. This indicates that speculators did not play a significant role in the auction. Second, similar items should sell for similar prices (cf., the Swiss and New Zealand auctions). Third, revenue should meet or exceed projections. Fourth, there should be evidence of price discovery — for example, prices and provisional winners at the mid-point of the auction should be highly correlated with final selling prices and winners. Finally, the packages assembled by bidders should be sensible, such as groups of licenses that are adjacent geographically or in frequency range.

SAAs have two big vulnerabilities. The first problem is *demand reduction*, and this is relevant even when items are substitutes. Demand reduction occurs when a bidder asks for fewer items than it really wants, to lower competition and therefore the prices paid for the items that it gets.

To illustrate, suppose there are two identical items and two bidders. The first bidder has

valuation 10 for one of the items and valuation 20 for both. The second bidder has valuation 8 for one of the items and does not want both (i.e., its valuation remains 8 for both). Giving both items to the first bidder maximizes the welfare, at 20. The VCG mechanism would earn revenue 8 on this example. Now consider how things play out in an SAA. Bidder 2 would be happy to have either item at any price less than 8. Thus, bidder 2 drops out only when both items have price at least 8. If bidder 1 stubbornly insists on winning both items, its utility will be $20 - 16 = 4$. If, on the other hand, bidder 1 targets just one item, then each of the bidders will get one of the items at a near-zero price. Bidder 1's utility is then close to 10. In this example, demand reduction leads to a loss of welfare and revenue, relative to the VCG mechanism's outcome. There is ample evidence of demand reduction in many spectrum auctions.

The second big problem with SAAs, which is relevant when items are complements (including in many spectrum auctions), is the *exposure problem*. As an example, consider two bidders and two non-identical items. Bidder 1 only wants both items — they are complementary items for the bidder — and its valuation is 100 for them (and 0 otherwise). Bidder 2 is willing to pay 75 for either item. The VCG mechanism would give both items to bidder 1, for a welfare of 100, and would generate revenue 75. In a SAA, though, bidder 2 will not drop out until the price of each item reaches 75. Bidder 1 is in a no-win situation: to get both items it would have to pay 150, more than its value. The scenario of winning only one item for a non-trivial price could be even worse. On the other hand, if bidder 2's value for each item was only 40, then bidder 1 should just go for it. But how can bidder 1 know which scenario is closer to the truth? The exposure problem makes bidding in an SAA difficult for a bidder for whom items are complements, and it often leads to risk-averse, tentative bidding by such bidders.

3 Bells and Whistles

A difficult and controversial question is whether or not to augment the basic SAA format by package bidding — bidding on sets of items in addition to individual items – and, if so, how. The primary reason to allow package bidding is to alleviate the exposure problem when items are complements, to free up bidders who desire bundles of items to bid aggressively for them. There are also scenarios where package bids can remove the incentive for demand reduction.

The conservative viewpoint, which dominated practice until relatively recently, is that package bids add complexity to a quite functional auction format and might do more harm than good. Limited forms of package bidding have been incorporated into spectrum auction designs only over the past 5–10 years.

One design approach is to tack on one extra "proxy" round after the SAA where bidders can submit package bids on any subsets of items that they want, subject to an activity rule; see Ausubel and Milgrom [1] for details. These package bids compete with each other as well as the winning bids on individual items from the SAA phase of the auction. The final allocation is determined by a welfare-maximization computation, treating bids as true

(a) Before (b) After

Figure 1: After some of the TV broadcasters are bought out, the remaining ones will be repacked via channel reassignment to free up a contiguous portion of the spectum.

values. The biggest issue with this approach is that computing the final prices is tricky. The VCG payment rule is not used because of its poor revenue and incentive properties (see Lecture 7 and the exercises). A more aggressive payment rule, which is not DSIC but does have other good incentive properties, is used instead. Typical behavior of bidders with this relatively complex pricing rule does not seem to be well understood.

A second approach is to predefine a limited set of allowable package bids, rather than allowing bidders to propose their own. Ideally, the predefined package bids should be well-aligned with what bidders actually want, yet structured enough to permit reasonably simple allocation and payment rules. Hierarchical packages — for example, allowing bids on individual licenses, on regional bundles of licenses, and on nationwide bundles — have emerged as a sweet spot for this design approach [3]. The biggest issue with predefined package bids is that they can do more harm than good when they are poorly matched with bidders' goals. For example, imagine that you're a bidder who wants the items ABCD, but the available packages are ABEF and CDHI — what's your bidding strategy?

4 The Cutting Edge

We've reached the state-of-the-art of wireless spectrum auctions, so let's conclude with a peek into the future: an upcoming FCC double auction, to take place possibly in 2014.[2]

Wireless spectrum doesn't grow on trees. At this point, giving someone a new allocation of spectrum generally requires taking it away from someone else. Soon, the FCC plans to do precisely this, using a reverse auction (cf., Exercise 7) to free up spectrum by buying out TV broadcasters and then a forward auction to resell the spectrum to companies that can put it to more valuable use. The forward auction will likely be implemented as an SAA with bells and whistles, as usual; the reverse auction is completely new.

In addition, the FCC will repack the remaining broadcasters so that the freed up frequency is contiguous. For example, they might buy out a number of TV broadcasters across the nation who were using a UHF channel somewhere between 38 and 51, and reassign all of the remaining broadcasters to have a channel between 38 and 45, leaving the part of the spectrum corresponding to channels 46–51 free for new users (see Figure 1).

[2]The auction format is still under discussion, and this section is only the author's best guess as to what will be adopted.

In a very cool development, the current frontrunner for the reverse auction format is a greedy approximate welfare-maximizing allocation rule, not unlike those we discussed for Knapsack auctions in Lecture 4. In the proposed model, each bidder i (a TV broadcaster) has a private valuation v_i for its broadcasting license. That is, v_i is the "minimum acceptable offer" for buying out i.[3] Letting N denote the set of bidders, a set $S \subseteq N$ of winning bidders — where "winning" means being bought out — is feasible if the remaining bidders $N \setminus S$ can be repacked in the target range (e.g., channels 38–45).[4] For instance, if $S = N$ then all bidders are bought out and the entire spectrum is freed up, so S is certainly feasible. When $S = \emptyset$, no spectrum is freed up, an infeasible outcome. Checking whether or not a given set S is feasible is a medium-size NP-hard problem — essentially the graph coloring problem, since two TV stations with overlapping geographic areas cannot be assigned the same or adjacent channels — so solving it requires state-of-the-art algorithmic technology. As of this writing, SAT solvers and integer programming solvers are battling it out, striving to solve these "feasibility-checking" problems as fast as possible (ideally, in seconds).

We give a direct-revelation description of the proposed class of allocation rules, although they can (and likely will be) implemented via an iterative auction with descending, bidder-specific prices. Descending implementations are preferred to sealed-bid implementations because, empirically, bidders find them easier to play. The allocation rule starts with the trivial feasible set (all bidders), and then iteratively removes bidders from the current feasible set until a minimal feasible set is reached. A greedy scoring rule is used to choose which bidder to remove in each iteration. One might call this a "reverse greedy algorithm," since it deletes bidders starting from the entire set, rather than forward greedy algorithms which iteratively add bidders starting from the empty set (cf., Knapsack auctions in Lecture 4). Milgrom and Segal [5] call these *deferred allocation rules*.

- Set $S = N$. [Initially feasible.]
- While there is an $i \in S$ such that $S \setminus \{i\}$ remains feasible:
  - (*) Delete some such i from S. [I.e., i will not be bought out.]
- Return S.

Step (*) is obviously underdetermined, and it's easy to think of various heuristics to try, like deleting the bidder with the highest bid (i.e., least willing to be bought out), the bidder with

[3] This single-parameter model assumes that each TV station is owned by a different strategic agent. This assumption is not entirely true in practice, but it makes the model much easier to reason about.

[4] One interesting question is how to set this target. The bigger the target, the bigger the expenses per unit of spectrum in the reverse auction and the smaller the revenues per unit of spectrum in the forward auction, since increasing supply should decrease the price. An ideal target would equalize the price per spectrum in the forward and reverse auctions — or perhaps with a somewhat higher price in the forward auction, so that auction expenses are recovered by the auction's net revenue. One approach that is being discussed seriously is to use the proposed reverse auction to estimate the entire supply curve — the cost of acquiring spectrum for each possible target — and then match supply and demand accordingly during the forward auction.

the highest per-capita bid, etc. The exact choice of the greedy rule will likely be guided by the welfare achieved by different rules on synthetic data.

If we implement (*) using a scoring function, deleting the bidder i with the largest score (subject to $S \setminus \{i\}$ being feasible), and if this scoring function is increasing in a bidder's bid and independent of the bids of the other active players, then the deferred allocation rule is monotone (bidding lower can only cause you to win). See Exercise Set #4. By Myerson's Lemma, paying critical bids — the largest bid that a winning bidder could have made and still gotten bought out — yields a DSIC auction.

Remarkably, deferred allocation rules have a number of good incentive properties above and beyond DSIC, which are not shared by their forward-greedy cousins [5]; see also Problem Set #3.

References

[1] L. Ausubel and P. Milgrom. Ascending proxy auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, chapter 3. MIT Press, 2006.

[2] P. Cramton. Simultaneous ascending auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, chapter 4. MIT Press, 2006.

[3] J. K. Goeree and C. A. Holt. Hierarchical package bidding: A paper & pencil combinatorial auction. *Games and Economic Behavior*, 70(1):146–169, 2010.

[4] P.. Milgrom. *Putting Auction Theory to Work*. Cambridge, 2004.

[5] P. Milgrom and I. Segal. Deferred-acceptance heuristic auctions. Working paper, August 2013.

CS364A: Algorithmic Game Theory
Lecture #9: Beyond Quasi-Linearity*

Tim Roughgarden†

October 21, 2013

1 Budget Constraints

Our discussion so far has assumed that each agent has *quasi-linear utility*, meaning that it acts to maximize its valuation $v_i(\omega)$ for the chosen outcome ω minus the payment p_i that it has to make. Thus, a bidder's utility is a linear function of the payment made. We have placed no restrictions on payments, other than the minimal conditions that they are nonnegative and no more than the bid $b_i(\omega)$ agent i made for the chosen outcome.

In some important applications, payments are constrained. We first focus on *budget constraints*, which limit the amount of money that an agent can pay. Sometimes, there is little need to incorporate budget constraints. In a single-item auction, where we interpret the valuation of an agent as its maximum willingness-to-pay, its valuation is presumably bounded above by its budget. In other applications, especially where an agent might wind up buying a large number of items, budgets are crucial.

For example, every keyword auction used in practice asks a bidder for its bid-per-click (e.g., \$.25) and its daily budget (e.g., \$100). Per-item values and overall budgets model well how many people made decisions in auctions with lots of items, especially when the items are identical.

The simplest way to incorporate budgets into our existing utility model is to redefine the utility of player i with budget B_i for outcome ω and payment p_i as

$$\begin{array}{rl} v_i(\omega) - p_i & \text{if } p_i \leq B_i \\ -\infty & \text{if } p_i > B_i. \end{array}$$

One can of course study smoothed version of this utility function, where there is a cost that is an increasing function of the budget violation.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

Payment constraints such as budget constraints join the two other types of constraints we've been operating under along: incentive constraints, often in the form of monotonicity; and allocation constraints, such as allocating each good to at most one agent. Surplus maximization, where payments appear neither in the objective function nor in the constraints (other than being between 0 and bidders' bids), is special. The VCG mechanism, and its precursors in Lectures 2–4, maximizes surplus "ex post," meaning as well as if all of the private data is known a priori. Maximizing revenue, where payments participate in the objective function, requires new auction formats and new measures of success (Lectures 5 and 6). The same is true of mechanism design with payment constraints, the subject of this and the next lecture.

We certainly can't maximize the surplus $\sum_{i=1}^{n} v_i(\omega)$ ex post when there are budget constraints. Consider the simple case of a single-item auction, where every bidder has a known budget of 1 and a private valuation.[1] The Vickrey auction charges the winner the second-highest bid, which might well be more its budget. Since the Vickrey auction is the unique DSIC surplus-maximizing auction (Exercise 9), surplus-maximization is impossible without violating budgets. As shown in the exercises, no DSIC auction that respects budgets can approximate the surplus well. We need new auction formats to accommodate budget constraints.

2 The Clinching Auction

The original clinching auction, by Ausubel [1], is an ascending implementation of the VCG mechanism when there are multiple identical items, analogous to the English auction for a single item. In [1], a bidder might want more than one item but is assumed to have nonincreasing marginal values for items. We discussed last lecture why ascending auctions can be more desirable than direct-revelation mechanisms. Unlike the SAA format discussed last lecture, the clinching auction in [1] is immune to demand reduction. See also Problem Set #3.

We discuss the variation of the clinching auction, due to Dobzinski et al. [3], that accommodates budget constraints. There are m identical goods, and each bidder might want many of them (like clicks in a keyword auction). Each bidder i has a private valuation v_i for each good that it gets — so if it gets k goods, its valuation for them is $k \cdot v_i$. Each bidder has a budget B_i that we assume is *public*, meaning it is known to the seller in advance.[2] The clinching auction described in this section is not DSIC when budgets are private (see the exercises).

[1] We argued that budgets are often superfluous in a single-item auction, but the point we're making here is general.

[2] We'd love to assume that budgets are private and thus also subject to misreport, but private budgets make the problem tougher, even impossible in some senses [3]. The version of the problem with public budgets is hard enough already — as shown above, surplus maximization ex post is impossible — and it guides us to some elegant and potentially useful auction formats, which of course is the whole point of the exercise.

2.1 First Cut: Using the Market-Clearing Price

We first describe an auction that is more naive than the clinching auction. One can view the clinching auction as a revised, more sophisticated version of this naive auction. We give a direct-revelation description; it will be clear that there is an ascending implementation of it.

The first auction is based on selling goods at the "market-clearing price", where supply equals demand. It's clear what the supply is (m, the number of goods). The demand of a bidder depends on the current price, with higher prices meaning less demand. Formally we define the *demand of bidder i at price p* as:

$$D_i(p) = \begin{cases} \min\left\{\lfloor \frac{B_i}{p} \rfloor, m\right\} & \text{if } p < v_i \\ 0 & \text{if } p > v_i. \end{cases}$$

To explain, recall that bidder i has value v_i for every good that it gets. If the price is above v_i it doesn't want any (i.e., $D_i(p) = 0$), while if the price is below v_i it wants as many as it can afford (i.e., $D_i(p) = \lfloor \frac{B_i}{p} \rfloor$). When $v_i = p$ the bidder does not care how many goods it gets, as long as its budget is respected, and in the auction and its analysis we can take $D_i(v_i)$ to be a convenient integer in $\{0, 1, 2, \ldots, \lfloor \frac{B_i}{p} \rfloor\}$ of our choosing.

As the price p rises, demand $D_i(p)$ goes down, from $D_i(0) = m$ to $D_i(\infty) = 0$. A demand drop can have two different forms: from an arbitrary positive integer to 0 (when p hits v_i), or by a single unit (when $\lfloor B_i/p \rfloor$ becomes one smaller).

Let p^* be the smallest price with $\sum_i D_i(p^*) = m$. Or, more generally, the smallest value such that $\lim_{p \uparrow p^*} \sum_i D_i(p) \geq m \geq \lim_{p \downarrow p^*} \sum_i D_i(p)$. Then, the auction gives $D_i(p^*)$ goods to each bidder i, each at the price p^* (defining $D_i(p^*)$'s for bidders i with $v_i = p^*$ so that all m goods are allocated).

The good news is that, by the definition of the demand $D_i(p)$, this auction respects bidders' budgets. The bad news is that it is not DSIC; it is vulnerable to demand reduction, similar to the simultaneous ascending auction format discussed last lecture.

Example 2.1 (Market-Clearing Price Is Not DSIC) Suppose there are two goods and two bidders, with $B_1 = +\infty$, $v_1 = 6$, and $B_2 = v_2 = 5$. First suppose that both bidders bid truthfully. The total demand $\sum_i D_i(p)$ is at least 3 until the price hits 5, at which point $D_1(5) = 2$ and $D_2(5) = 0$. The auction thus allocates both goods to bidder 1 at a price of 5 each, for a utility of 2. If bidder 1 falsely bids 3, however, it does better. The reason is that bidder 2's demand drops to 1 at the price $\frac{5}{2}$ (it can no longer afford both), and the auction will terminate at the price 3, at which point $D_1(3)$ will be defined as 1. Bidder 1 only gets one good, but the price is only 3, so its utility is 3, more than with truthful bidding.

We haven't studied any non-DSIC auctions since Myerson's Lemma (Lecture 3), which in some sense gives a complete solution to DSIC auction design in single-parameter settings like the present one. The allocation rule in the market-clearing price auction is monotone, as you are invited to check, so Example 2.1 shows that we got the payment rule wrong. We could apply Myerson's Lemma to this allocation rule to derive the appropriate payments to recover DSIC, but we'll want a slightly more sophisticated allocation rule, as well.

2.2 The Clinching Auction for Bidders with Budgets

We'll again give a direct-revelation description, but keep in mind that the auction admits a natural ascending implementation, and that this was the original point of the clinching auction.

Rather than sell all the goods in one shot, we will sell them piecemeal, at different prices. In addition to the current price p, the auction keeps track of the current supply s (initially m) and the residual budget $\hat{B}_i$ (initially B_i) of each bidder i. The demand $\hat{D}_i(p)$ of bidder i at price p is defined with respect to the residual budget and supply, as $\min\{\lfloor \frac{\hat{B}_i}{p} \rfloor, s\}$ if $p < v_i$ and as 0 if $p > v_i$.

Clinching Auction for Budgeted Bidders

- Initialize $p = 0$, $s = m$.
- While $s > 0$:
  - Increase p until there is a bidder i such that $\underbrace{s - \sum_{j \neq i} \hat{D}_j(p)}_{:=k} > 0$.
  - Give k goods to bidder i at price p (theses good are "clinched").
  - Decrease s by k.
  - Decrease $\hat{B}_i$ by $p \cdot k$.

Observe that different goods are sold at different prices, with selling prices increasing over the course of the auction. Observe also that budgets are respected — equivalently, the number of goods k clinched by a bidder i is at most its current demand $\hat{D}_i(p)$.[3]

Example 2.2 Let's return to the setting of Example 2.1 — two goods and two bidders, with $B_1 = +\infty$, $v_1 = 6$, and $B_2 = v_2 = 5$. Suppose both bidders bid truthfully. In Example 2.1, bidder 1 was awarded both goods at a price of 5. Here, because the demand $D_2(p)$ of the second bidder drops to 1 once $p = \frac{5}{2}$, bidder 1 clinches one good at a price of $\frac{5}{2}$. The second good is sold to bidder 1 at price 5, as before. Thus bidder 1 has utility $\frac{9}{2}$ when it bids truthfully in the clinching auction. As we'll see, no false bid could be better.

Theorem 2.3 *The clinching auction for bidders with public budgets is DSIC.*

Proof: We could proceed by verifying that the allocation rule is monotone and the payments conform to Myerson's payment formula, but it's easier to just verify the DSIC condition directly. So, fix a bidder i and bids $\mathbf{b}_{-i}$ by the others. Since bidder i's budget is public, it cannot affect the term $\lfloor \hat{B}_i/p \rfloor$ of its demand function $\hat{D}_i(p)$. It can only affect the time at which it is kicked out of the auction (meaning $\hat{D}_i(p) = 0$ forevermore), which is precisely

[3] If not, then $\sum_j \hat{D}_j(p) < s$. But the auction maintains the invariant that the sum of the current demands is at least the current supply.

when the price p reaches its bid b_i. Note that every good clinched by bidder i when $p < v_i$ contributes positively to the bidder's utility, while every good clinched when $p > v_i$ contributes negatively.

First compare the utility earned by bid $b_i < v_i$ to that earned by a truthful bid. Imagine running the clinching auction twice in parallel, once when i bids b_i and once when i bids v_i. By induction on the number of iterations, the execution of the clinching auction will be identical in the two scenarios as the price ascends from 0 to b_i. Thus, by bidding b_i, the bidder can only lose out on goods that it otherwise would have cliched (for nonnegative utility) in the price interval $[b_i, v_i]$.

Similarly, if i bids $b_i > v_i$, all that changes is that the bidder might acquire some additional goods for nonpositive utility in the price interval $[v_i, b_i]$. Thus, no false bid nets i more utility than a truthful one. ■

If budgets are private and the clinching auction is run with reported budgets instead, then it is no longer DSIC (see the exercises).

Taken alone, Theorem 2.3 is not compelling. There are other simple budget-respecting DSIC auctions, such as giving away all the goods to random bidders for free. We would like to additionally say that the clinching auction computes a "good" allocation, such as one with surplus close to the maximum possible (subject to budget constraints). The original clinching auction [1], without budgets, implements the VCG outcome and hence is surplus-maximizing. As we've seen, no budget-respecting mechanism can have surplus close to that of the VCG mechanism (which need not respect budgets).

Researchers have explored at least three approaches to justifying the clinching auction with budgets on surplus grounds. None are fully satisfying. While there is strong belief that the clinching auction is "the right solution," researchers are struggling to formulate a model to make this intuition precise.

The key challenge is to identify a good benchmark to compare to the performance of the clinching auction. Dobzinski et al. [3] study Pareto optimality rather than an objective function. An allocation is *Pareto optimal* if and only if there's no way to reassign goods and payments to make some agent (a bidder or the seller) better off without making another worse off, where the seller's utility is its revenue. The good news is that Pareto optimality strongly advocates for the clinching auction — it is the unique deterministic DSIC auction that always computes a Pareto optimal allocation. The bad news is that Pareto optimality is not always necessary or sufficient for an auction to be desirable. For example, Bayesian-optimal mechanisms, discussed below, need not be Pareto optimal.

The second approach is to posit a distribution over bidders' valuations and solve for the DSIC mechanism that maximizes expected surplus subject to the given budget constraints (cf., Lecture 5). With this average-case approach, there is an unambiguous notion of "optimal" auctions — those with the highest expected surplus. It is also interesting to prove "simple near-optimal" and "prior-independent" approximations in this setting, along the lines of the results in Lecture 6. Progress in these directions have been slow but steady [6]. Common budgets are currently better understood than general budgets, and in this special case the clinching auction is provably near-optimal [2].

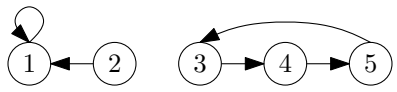


Figure 1: An iteration of the Top Trading Cycle Algorithm (TTCA) with two directed cycles.

A third approach is to modify the surplus objective function to take budgets into account. The most common proposal is to replace $\sum_i v_i x_i$ by $\sum_i \min\{B_i, v_i x_i\}$. The good news is that the clinching auction is provably near-optimal with respect to this objective function [4]. The bad news is that this modified objective does not make much sense in some settings; see the exercises and [5, §3.10].

3 Mechanism Design without Money

There are a number of important applications where there are significant incentive issues but where money is infeasible or illegal. This is equivalent to all agents having a budget of zero. Mechanism design without money is relevant for designing and understanding methods for voting, organ donation, school choice, and labor markets. The designer's hands are tied without money — even tighter than with budget constraints. There is certainly no Vickrey auction, for example. Despite this, and strong impossibility results in general settings, some of mechanism design's greatest hits are motivated by applications without money.

Shapley and Scarf [7] defined the following *house allocation problem*. There are n agents, and each initially owns one house. Each agent has a total ordering over the n houses, and need not prefer their own over the others. The question is: how to sensibly reallocate the houses to make the agents better off? Consider the following *Top Trading Cycle Algorithm (TTCA)*, credited to Gale in [7].

- While agents remain:
  - Each remaining agent points to its favorite remaining house. This induces a directed graph G on the remaining agents in which every vertex has out-degree 1 (Figure 1).
  - The graph G has at least one directed cycle.[4] Self-loops count as directed cycles.
  - Reallocate as suggested by the directed cycles, with each agent on a directed cycle C giving its house to the agent that points to it, that is, to its predecessor on C.
  - Delete the agents and the houses that were reallocated in the previous step.

Observe that the TTCA terminates with each agent possessing exactly one house. As a sanity check for its reasonableness, observe that every agent is only made better off by the algorithm. To see why, note that the algorithm maintains the invariant that the remaining agents still own their original houses. Thus, every iteration, an agent points either to its

[4]Keep following outgoing arcs; eventually, a vertex will be repeated, exposing a directed cycle.

own house or to a house that it likes better. Finally, when an agent is deleted, it receives the house that it had been pointing to.

When agents' preferences are privately known, we can apply the TTCA to agents' reported preferences in a direct-revelation mechanism. There is no incentive for agents to misreport their preferences.

Theorem 3.1 *The TTCA induces a DSIC mechanism.*

Proof: Let N_j denote the agents allocated in the jth iteration of the TTCA when all agents report truthfully. Each agent of N_1 gets its first choice and hence has no incentive to misreport. An agent i of N_2 is not pointed to by any agent of N_1 in the first iteration — otherwise, i would belong to N_1 rather than N_2. Thus, no misreport by i nets a house originally owned by an agent in N_1. Since i gets its first choice outside of the houses owned by N_1, it has no incentive to misreport. In general, an agent i of N_j is never pointed to in the first $j-1$ iterations of the TTCA by any agents in $N_1 \cup \cdots \cup N_{j-1}$. Thus, whatever it reports, i will not receive a house owned by an agent in $N_1 \cup \cdots \cup N_{j-1}$. Since the TTCA gives i its favorite house outside this set, it has no incentive to misreport. ■

As with the clinching auction, Theorem 3.1 by itself is not impressive — the mechanism in which every agent keeps its initial house is also DSIC. To argue that the TTCA is in some sense optimal, we introduce the notion of a *core allocation* — an allocation such that no coalition of agents can make all of its members better off via internal reallocations.

Theorem 3.2 *For every house allocation problem, the allocation computed by the TTCA is the unique core allocation.*

Proof: To prove the computed allocation is a core allocation, consider an arbitrary subset S of agents. Define N_j as in the proof of Theorem 3.1. Let ℓ be the first iteration in which $N_\ell \cap S \neq \emptyset$, with agent $i \in S$ receiving its house in the ℓth iteration of TTCA. TTCA gives agent i its favorite house outside of those owned by $N_1, \ldots, N_{\ell-1}$. Since no agents of S belong to $N_1, \ldots, N_{\ell-1}$, no reallocation of houses among agents of S can make i strictly better off.

We now prove uniqueness. In the TTCA allocation, all agents of N_1 receive their first choice. This must equally be true in any core allocation — in an allocation without this property, the agents of N_1 that didn't get their first choice form a coalition for which internal reallocation can make everyone strictly better off. Similarly, in the TTCA allocation, all agents of N_2 receive their first choice outside of N_1. Given that every core allocation agrees with the TTCA allocation for the agents of N_1, such allocations must also agree for the agents of N_2 — otherwise, the agents of N_2 that fail to get their first choice outside N_1 can all improve via an internal reallocation. Continuing inductively, we find that the TTCA allocation is the unique core allocation. ■

References

[1] L. M. Ausubel. An efficient ascending-bid auction for multiple objects. *American Economic Review*, 94(5):1452–1475, 2004.

[2] N. R. Devanur, B. Q. Ha, and J. D. Hartline. Prior-free auctions for budgeted agents. In *Proceedings of the 14th ACM Conference on Electronic Commerce (EC)*, pages 287–304, 2013.

[3] S. Dobzinski, R. Lavi, and N. Nisan. Multi-unit auctions with budget limits. *Games and Economic Behavior*, 74(2):486–503, 2012.

[4] S. Dobzinski and R. Paes Leme. Efficiency guarantees in auctions with budgets. In *Ninth Ad Auctions Workshop*, 2013.

[5] N. Nisan, J. Bayer, D. Chandra, T. Franji, R. Gardner, Y. Matias, N. Rhodes, M. Seltzer, D. Tom, H. R. Varian, and D. Zigmond. Google's auction for TV ads. In *36th Internatilonal Colloquium on Automata, Languages and Programming (ICALP)*, pages 309–327, 2009.

[6] M. M. Pai and R. Vohra. Optimal auctions with financially constrained buyers. To appear in *Journal of Economic Theory*, 2013.

[7] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

CS364A: Algorithmic Game Theory
Lecture #10: Kidney Exchange and Stable Matching*

Tim Roughgarden†

October 23, 2013

1 Case Study: Kidney Exchange

Many people suffer from kidney failure and need a kidney transplant. Currently, the US waiting list for kidneys has about 100,000 people on it. An old idea, used also for other organs, is deceased donors — when someone dies and is a registered organ donor, their organs can be transplanted into others. One special feature of kidneys is that a healthy person has two of them and can survive just fine with only one of them. This creates the possibility of *living* organ donors, such as a family member of the patient in need.

Unfortunately, having a living kidney donor is not always enough — sometimes a patient-donor pair is *incompatible*, meaning that the donor's kidney is unlikely to function well in the patient. Blood and tissue types are the primary culprits for incompatibilities. For example, a patient with O blood type can only receive a kidney from a donor with the same blood type, and similarly an AB donor can only donate to an AB patient.

Suppose patient P1 is incompatible with its donor D1 because they have blood types A and B, respectively. Suppose P2 and D2 are in the opposite boat, with blood types B and A, respectively (Figure 1). Even though (P1,D1) may never have met (P2,D2), exchanging donors seems like a pretty good idea — P1 can get its kidney from D2 and P2 from D1. This is called a *kidney exchange*.

A few kidney exchanges were done, on an ad hoc basis, around the beginning of this century. These isolated successes make clear the need for a nationwide kidney exchange, where incompatible patient-donor pairs can register and be matched with others. How should such an exchange be designed? The goal of such an exchange is to thicken the kidney exchange market to enable as many matches as possible.

National kidney exchanges sprang up around the middle of last decade. We'll cover some of the early design ideas for these exchanges, as well as current challenges. These exchanges

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

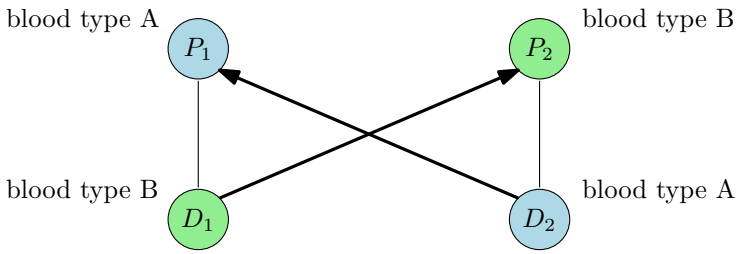


Figure 1: A kidney exchange

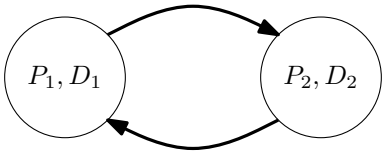


Figure 2: A good case for the Top Trading Cycle Algorithm.

and their underlying algorithms have been quite successful, enabling thousands of successful kidney transplants every year.

Currently, compensation for organ donation is illegal in US (and in every country except for Iran). Kidney exchange is legal. Thus kidney exchange is an ideal application for mechanism design without money, for whatever incentive issues are relevant (and there are a few). It's interesting to speculate about whether or not a monetary market for kidneys will exist in any Western countries in, say, 10 years. (Iran does not have a waiting list for kidneys.)

1.1 Idea #1: Use the Top Trading Cycle Algorithm

This section and the next cover two relatively early papers by Roth, Sönmez, and Ünver [5, 6] that are influential brainstorms about what an incentive-compatible national kidney exchange might look like. In the first paper [5], which covers research done before the authors talked extensively to doctors, advocated the Top Trading Cycle Algorithm (TTCA) from last lecture as a starting point. In its most basic form, the correspondence is between agent-initial house pairs in the housing problem and patient-donor pairs in the kidney exchange problem — the initial house of an agent corresponds to its incompatible living donor. The housing problem assumes that each agent has a total ordering over houses; in kidney exchange, it is natural to order donors according to the estimated probability that their kidney could be successfully transplanted into the patient (based on blood type, tissue type, etc.).

The goal of applying the TTCA is to find cycles like that in Figure 2, where with patient-donor pairs as in Figure 1, each patient points to the other's donor as its favorite. Reallocating donors according to this cycle is the favorable kidney exchange identified earlier. More generally, donors are reallocated to patients so that everyone is collectively as well off as possible (in the senses discussed last lecture).

The actual kidney exchange problem is more complicated in several ways. We first discuss one extension that can be accommodated by the TTCA approach, and then two

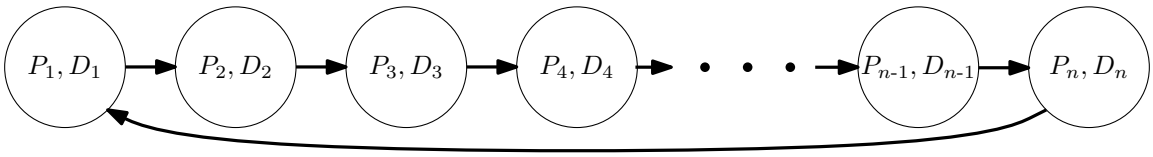


Figure 3: A bad case for the Top Trading Cycle Algorithm.

challenges that motivate reformulating the problem. The extension of the TTCA in [5] handles, in addition to incompatible patient-donor pairs, patients without living donors (an agent without an initial house) and deceased donors (houses without an initial owner). The new algorithm allocates along cycles as in the TTCA, and also along suitable chains (i.e., paths). Provided chains are chosen in the right way, the algorithm remains DSIC — so no patient (or patient's doctor) can misreport information to increase the estimated probability of a successful transplant.[1]

The next issue with the TTCA algorithm is a dealbreaker in the context of kidney exchange: the cycles along which reallocations are made can be arbitrarily long. For example, in the first iteration of TTCA, it is possible that the arcs corresponding to preferred donors form a Hamiltonian cycle on patient-donor pairs (Figure 3).

Why are long cycles a problem? Consider first a cycle of length two (Figure 2). Note this already corresponds to four surgeries — two to extract donors' kidneys, and two to implant them in the patients. Moreover, these four surgeries *must happen simultaneously*. Incentives are the reason: in the example in Figure 1, if the surgeries for P1 and D2 happen first, there is a risk that D1 will renege on its offer to donate its kidney to P2. One problem is that P1 unfairly got a kidney "for free;" the much more serious problem is that P2 is as sick as before and, since its donor D2 has donated its kidney, P2 can no longer participate in a kidney exchange. Because of this risk, no one wants to experiment with non-simultaneous surgeries in kidney exchange.[2] The constraint of simultaneous surgeries — with each surgery needing its own operating room, surgical team, etc. — motivates keeping reallocation cycles as small as possible.

Our final critique of the TTCA approach is that modeling preferences as a total ordering over the set of living donors is overkill: empirically, patients don't really care which kidney they get as long as it is compatible with them. Binary preferences over donors are therefore more appropriate.

[1]This extension was largely worked out earlier in a different context: allocating dorm rooms to students when there is a mixture of incumbents (students who already have a room but might want to upgrade), empty rooms (e.g., from students who graduated), and students without a current room (e.g., new students) [1].

[2]Sequential surgeries are now being used in slightly different contexts. For example, there is a handful of altruistic living donors, who are interested in donating a kidney even though they don't personally know anyone who needs one. An altruistic living donor can be the start of a chain of reallocations. Chains as long as 30 have been implemented [7], and at this scale surgeries have to be sequential. While the first problem of sequential surgeries (getting a kidney "for free") persists in this setting, the much more serious second problem (losing your own living donor) cannot occur in chains initiated by an altruistic living donor.

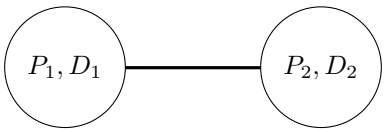


Figure 4: Applying a matching algorithm.

1.2 Idea #2: Use a Matching Algorithm

The second approach [6], motivated by the twin goals of binary preferences and short reallocation cycles, is to use *matchings*. Recall that a matching of an undirected graph (V, E) is a subset of edges $F \subseteq E$ such that no two edges of F share an endpoint.

The relevant graph for kidney exchange has a vertex set V corresponding to incompatible patient-donor pairs (one vertex per pair), and an undirected edge between vertices $(P1, D1)$ and $(P2, D2)$ if and only if P1 and D2 are compatible and P2 and D1 are compatible. Thus, the example in Figure 1 corresponds to the simple undirected graph in Figure 4. We define the optimal solutions to be the matchings of this graph that have maximum cardinality — that is, we want to arrange as many compatible kidney transplants as possible. By restricting the feasible set to matchings of this graph, we are restricting to pairwise kidney exchanges, and hence "only" 4 simultaneous surgeries, like in Figure 1.[3]

Our model for agents is that each vertex i has a true set E_i of incident edges, and can report any subset $F_i \subseteq E_i$ to a mechanism. Proposed kidney exchanges can be refused by a patient for any reason, so one way to implement a misreport is to refuse exchanges in $E_i \setminus F_i$. All that a patient cares about is being matched to a compatible donor. Our mechanism design goal is to compute an optimal solution (i.e., a maximum-cardinality matching) and to be DSIC, meaning that for every agent, reporting its full edge set is a dominant strategy.

Given our design goals, the mechanism must have the following form.

(1) Collect a report F_i from each agent i.

(2) Form the edge set$E = \{(i, j) : (i, j) \in F_i \cap F_j\}$. That is, include edge (i, j) if and only if both endpoints agree to the exchange.

(3) Return a maximum-cardinality matching of the graph $G = (V, E)$, where V is the (known) set of patient-donor pairs.

The mechanism is not yet fully specified, since there can be multiple choices for a maximum matching in step (3). Getting the tie-breaking right is important for achieving DSIC. There are two senses in which the maximum matching of a graph can fail to be unique. First, different sets of edges can be used to match the same set of vertices; see Figure 5. Since a

[3]These days, 3-way exchanges, corresponding to a directed cycle of 3 patient-donor pairs (with D2 compatible with P1, D3 with P2, and D1 with P3), are increasingly common. The reason is that 3-way exchanges are still logistically feasible and allowing them significantly increases the number of patients that can be saved. Empirically, exchanges involving 4 or more pairs don't really help match more patients, so they are not typically done.

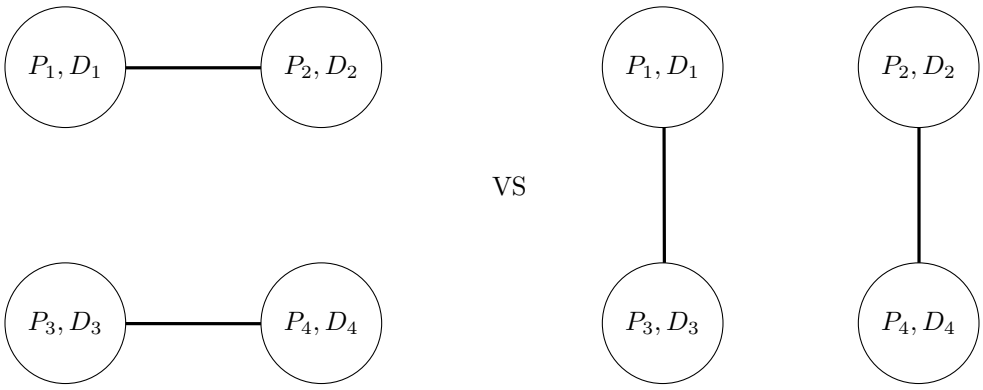


Figure 5: Different matchings can match the same set of vertices.

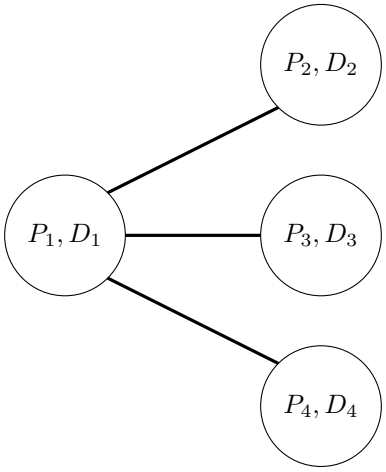


Figure 6: Different maximum matchings can match different subsets of vertices.

vertex does not care whom it is matched to, as long as it is matched, there is no reason to distinguish between different matchings that match the same set of vertices. More significantly, different maximum-cardinality matchings can match different subsets of vertices. For example, in the star (Figure 6), vertex 1 is in every maximum matching, but one and only one of the spokes will be chosen. How should we choose which one to return?

One solution is to prioritize the vertices before the mechanism begins. Without loss of generality, assume that the vertices $1, 2, \ldots, n$ are ordered from highest to lowest priority.[4] Then, we implement step (3) as follows:

(3a) Let M_0 denote the set of maximum matchings of G.

(3b) For $i = 1, 2, \ldots, n$:

[4]One appeal of this approach is that most hospitals already rely on similar priority schemes to manage their patients. The priority of a patient on a waiting list is determined by numerous factors, such as the length of time it has been waiting on the list, the difficulty of finding a compatible kidney, etc.

(3b.i) Let Z_i denote the matchings in M_{i-1} that match vertex i.

(3b.ii) If $Z_i \neq \emptyset$, set $M_i = Z_i$.

(3b.iii) Otherwise, set $M_i = M_{i-1}$.

(3c) Return an arbitrary matching of M_n.

That is, in each iteration i, we ask if there is a maximum matching that respects previous commitments and also matches vertex i. If so, then we additionally commit to matching i in the final matching. If previous commitments preclude matching i in a maximum-cardinality matching, then we skip i and move on to the next vertex. By induction on i, M_i is a non-empty subset of the maximum matchings of G. Every matching of M_n matches the same set of vertices — the vertices i for which Z_i was non-empty — so the choice of matching in step (3c) is irrelevant.

Theorem 1.1 *For every collection $\{E_i\}_{i=1}^n$ of edge sets and every ordering of the vertices, the priority matching mechanism above is DSIC: no agent can go from unmatched to matched by reporting a strict subset F_i of E_i rather than E_i.*

We leave the proof to you; see the exercises.[5]

1.3 Cutting-Edge Challenges

Current research is focused on incentive problems at the *hospital* level, rather than at the level of individual patient-donor pairs. This change is well motivated because many patient-donor pairs are reported to national kidney exchanges by hospitals, rather than by the pairs themselves. The objectives of a hospital (to match as many of its patients as possible) and of society (to match as many patients overall as possible) are not perfectly aligned. The key incentive issues are best explained through examples.

Example 1.2 (The Need for Full Reporting) Suppose there are two hospitals, each with three patients (Figure 7). Edges represent patient-donor pairs that are mutually compatible, as with the matching mechanism above. Each hospital has a pair of patient-donor pairs that it could match internally, without bothering to report them to a national exchange. We definitely don't want the hospitals to "greedily" execute these internal matches in this example — if H_1 matches 1 and 2 internally and only reports 3 to the exchange, and H_2 matches 5 and 6 internally and only reports 4 to the exchange, then the exchange can't match 3 and 4 so no further matches are gained. By contrast, if H_1 and H_2 both report their full sets of 3 patient-donor pairs to the national exchange, then 1, 2, and 3 can be matched with 4, 5, and 6, respectively, and so all patients get matched. In general, the goal is to incentive hospitals to report all of their patient-donor pairs, to save as many lives as possible.

[5]The keen reader is encouraged to read the deeper exploration in [6] of how the maximum matching selected varies with the chosen priority ordering over vertices. There is a very clean answer that goes through the Gallai-Edmonds decomposition, a beautiful result from matching theory that characterizes the structure of maximum-cardinality matchings in undirected graphs.

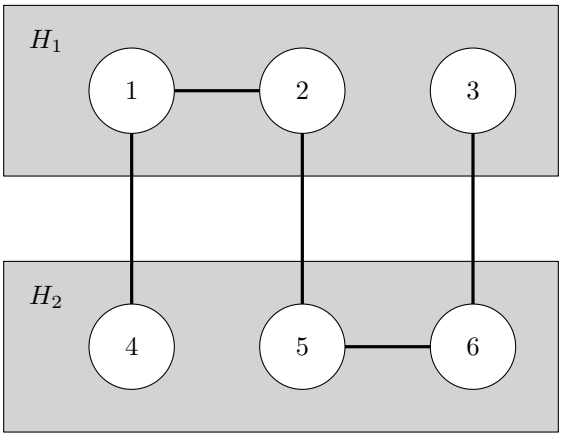


Figure 7: Example 1.2. Full reporting by hospitals leads to more matches than with only internal matches.

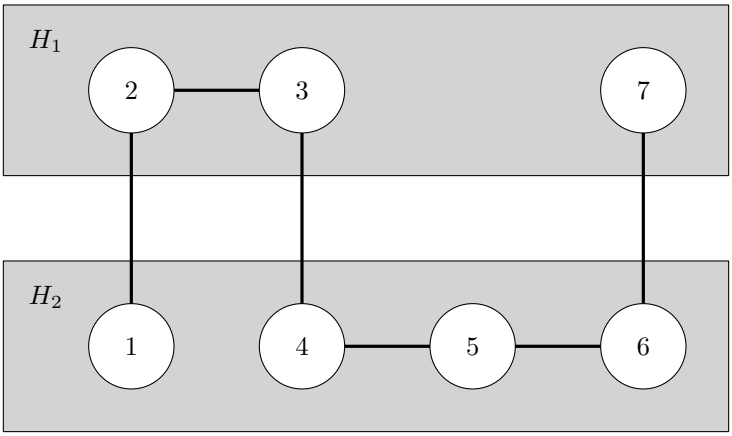


Figure 8: Example 1.3. Hospitals can have an incentive to hide patient-donor pairs.

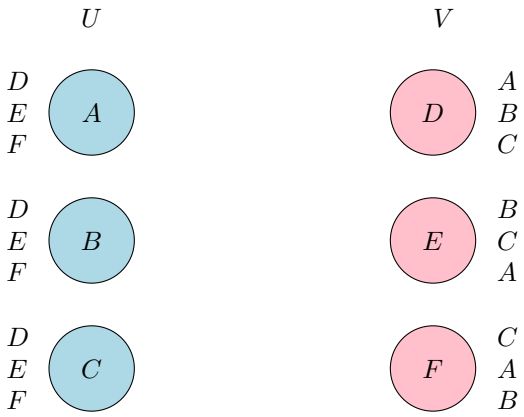


Figure 9: An instance of stable matching

Example 1.3 Consider again two hospitals, but now with 7 patients (Figure 8). Observe that, with an odd number of vertices, every matching leaves at least one patient unmatched. However, if H_1 hides patients 2 and 3 from the exchange (while H_2 reports truthfully), then H_1 guarantees that all of its patients are matched. The unique maximum matching in the report graph matches patient 6 with 7 (and 4 with 5), and H_1 can match 2 and 3 internally. On the other hand, if H_2 hides patients 5 and 6 while H_1 reports truthfully, then all of H_2's patients are matched. In this case, the unique maximum matching in the graph of report matches patient 1 with 2 and 4 with 3, while H_2 can match patients 5 and 6 internally.

The upshot is that there is irreconcilable tension between social and hospital incentives: there cannot be a DSIC mechanism that always computes a maximum-cardinality matching in the full graph.

In light of Example 1.3, the revised goal should be to compute an approximately maximum-cardinality matching so that, for each participating hospital, the number of its patients that get matched is approximately as large as in any matching, maximum-cardinality or otherwise. Understanding the extent to which this is possible, in both theory and practice, is an active research topic [2, 8].

2 Stable Matching

Stable matching is the canonical example of mechanism design without money. Killer applications include assigning medical school graduates to hospitals and students to elementary schools. The following model and algorithm are directly useful for these and other applications with amazingly few modifications.

We consider two sets U and V — the "men" and "women" — with n vertices each. Each vertex also has a total ordering over the vertices of the other set. For example, in Figure 9, the men all agree on the ranking of the women, while the women have very different opinions about the men.

Definition 2.1 A *stable matching* is a perfect bipartite matching — i.e., each vertex is matched to precisely one vertex from the other set — so that there is no "blocking pair," meaning:

(*) if $u \in U, v \in V$ are *not* matched, then either u prefers its mate v' in the matching to v, or v prefers its mate u' in the matching to u.

A perfect matching that failed condition (*) would spell trouble, since the blocking pair u and v would be tempted to run off with each other.[6]

We next discuss the famous proposal algorithm for computing a stable matching. Gale and Shapley [3] formalized the stable matching problem and gave this algorithm. Incredibly, it was later discovered that essentially the same algorithm had been used, since the 1950s, to assign medical residents to hospitals (as it is today) [4]!

The Proposal Algorithm:

- While there is an unattached man $u \in U$:
  - u proposes to its favorite woman who has not rejected him yet.
  - Each woman entertains only her best offer (from her perspective) thus far.

Example 2.2 Consider the instance in Figure 9. Suppose in the first iteration we choose the man C, who promptly proposes to his first choice, D. Woman D accepts because she currently has no other offers. If we pick man B in the next iteration, he also proposes to the woman D. Since woman D prefers B to C, she rejects C in favor of B. If we pick man A next, the result is similar: D rejects B in favor of A. A possible trajectory for the rest of the algorithm is: man C now proposes to his second choice, E; man B then also proposed to E, and E then rejects C in favor of B; and finally, C proposes to his last choice F, who accepts.

Several comments about the algorithm and its properties. First, it is underdetermined, leaving open how the unattached man is chosen. Second, note that each man systematically goes through his preference list, from top to bottom. Third, the men to which a woman is engaged only improve over the course of the algorithm. Fourth, the algorithm maintains the invariant that each man is matched to at most one woman and each woman is matched to at most one man.

Stable matchings and the proposal algorithm have an astonishing number of elegant properties. We begin with the most basic ones.

Theorem 2.3 (Computation of a Stable Matching) *The Proposal Algorithm terminates in at most n^2 iterations with a stable matching.*

[6] There is a clear similarity to the idea of a core allocation, introduced last lecture in the context of the housing allocation problem. If a matching is not stable, then it is not in the core — the pair u, v for which condition (*) is violated would be better off seceding from the mechanism. It's not hard to prove that the core allocations — matchings such that no subset of vertices could secede and do better — are precisely the stable matchings.

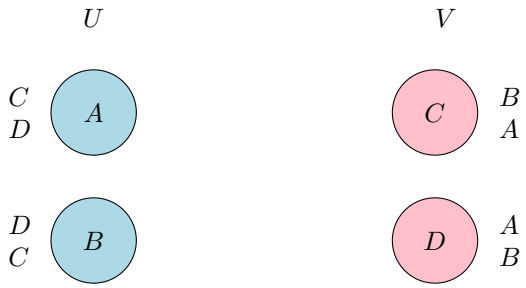


Figure 10: Example with multiple stable matchings

Corollary 2.4 (Existence of a Stable Matching) *For every collection of preference lists for the men and women, there exists at least one stable matching.*

We emphasize that Corollary 2.4 is *not* obvious a priori. Indeed, there are some simple variants of the stable matching problem for which a solution is not guaranteed.

Proof of Theorem 2.3: The bound on the number of iterations is easy to prove. Since each man works his way down his preference list, never proposing to the same woman twice, he makes at most n proposals. This makes for at most n^2 proposals (and hence iterations) overall.

Next, we claim that the proposal algorithm always terminates with a perfect matching. For if not, some man must have been rejected by all n women. A man is only rejected by a woman in favor of being matched to a better man, and once a woman is matched to a man, she remains matched to some man for the remainder of the algorithm. Thus, all n women must be matched at the end of the algorithm. But then all n men are also matched at the end of the algorithm, a contradiction.

Finally, we claim that the perfect matching computed is stable. To see why, consider a man $u \in U$ and a woman $v \in V$ who are not matched to each other. This can occur for two different reasons. In the first case, u never proposed to v. Since u worked its way down its list starting from the top, this means that u ended up matched to someone he prefers to v. If u did propose to v at some point in the algorithm, it must be that v rejected u in favor of a man she preferred (either at the time of u's proposal, or later). Since the sequence of men to which v is matched only improves over the course of the algorithm, she ended up matched to someone she prefers to u. ■

We noted earlier that the proposal algorithm does not specify how to choose among the unattached men in an iteration. Do all possible choices lead to the same stable matching? In Figure 9 there is only one stable matching, so in that example the answer is yes. In general, however, there can be more than one stable matching. In Figure 10, the men and the women both disagree on the ranking of the others. In the matching computed by the proposal algorithm, both men get their first choice, with A and B matched to C and D, respectively. Giving the women their first choices yields another stable matching.

We prove a stronger statement to resolve whether or not the output of the proposal algorithm is unique. For a man $u \in U$, let $h(u)$ be the highest-ranked woman (in u's

preference list) that u is matched to in *any* stable matching. Then:

Theorem 2.5 (Male-Optimal Stable Matching) *The stable matching computed by the proposal algorithm matches every man $u \in U$ to $h(u)$.*

Theorem 2.5 immediately implies that the output of the proposal algorithm is independent of which unattached man is chosen in each iteration. It also implies that there exists a "male-optimal" stable matching — a stable matching in which every man simultaneously attains his "best-case scenario" — no trade-offs between different men are necessary. A priori, there is no reason to expect the $h(u)$'s to be distinct and therefore induce a perfect matching.

Proof of Theorem 2.5: Let $R = \{(u, v)\}$ denote the pairs such that v rejects u at some point in a fixed execution of the proposal algorithm.

Since each man systematically works his way down his preference list, if u is matched to v at the conclusion of the algorithm, then $(u, v') \in R$ for every v' that u prefers to v. Thus, the following claim would imply the theorem: for every $(u, v) \in R$, *no* stable matching pairs up u and v.

We prove the claim by induction on the number of iterations of the proposal algorithm. Initially, $R = \emptyset$ and there is nothing to prove. For the inductive step, consider an iteration of the proposal algorithm in which v rejects u in favor of u' — thus one of u, u' proposed to v in this iteration.

Since u' systematically worked his way down his preference list, for every v' that u' prefers to v, (u', v') is already in the current set R of rejected proposals. By the inductive hypothesis, no stable matching pairs up u' with a woman he prefers to v — in every stable matching, u' is paired with v or someone less preferred. Since v' prefers u' to u, and u' prefers v to anyone else he might be matched to in a stable matching, there is no stable matching that pairs u with v (otherwise u', v would form a blocking pair). ■

It can also be shown that the proposal algorithm outputs the worst-possible stable matching from the perspective of the women — the algorithm matches each $v \in V$ to the lowest-ranked man $l(v) \in U$ that v is matched to in any stable matching.[7]

Suppose the preference lists of the vertices are private — is the proposal algorithm DSIC? That is, does a man or woman ever wind up with a better mate under misreported preferences than under truthfully reported preferences? As the male-optimality property of the computed stable matching might suggest, the proposal algorithm is DSIC for the men but not for the women. The former property is not trivial to prove (see Problem Set #3) while the latter can be verified in simple examples (see Exercise Set #5).

References

[1] A. Abdulkadiroğlu and T. Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88:233–260, 1999.

[7] Of course, one can change the algorithm to let the women make the proposals and the men the rejections to reverse both of these properties.

[2] I. Ashlagi, F. A. Fischer, I. A. Kash, and A. D. Procaccia. Mix and match: A strategyproof mechanism for multi-hospital kidney exchange. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 305–314, 2010.

[3] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[4] A. E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92:991–1016, 1984.

[5] A. E. Roth, T. Sönmez, and M. U. Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119:457–488, 2004.

[6] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125:151–188, 2005.

[7] K. Sack. 60 lives, 30 kidneys, all linked. *New York Times*, February 18 2012.

[8] P. Toulis and D. C. Parkes. A random graph model of kidney exchanges: Efficiency, individual-rationality and incentives. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, pages 323–332, 2011.

CS364A: Algorithmic Game Theory
Lecture #11: Selfish Routing and the Price of Anarchy*

Tim Roughgarden†

October 28, 2013

1 Quantifying the Inefficiency of Equilibria

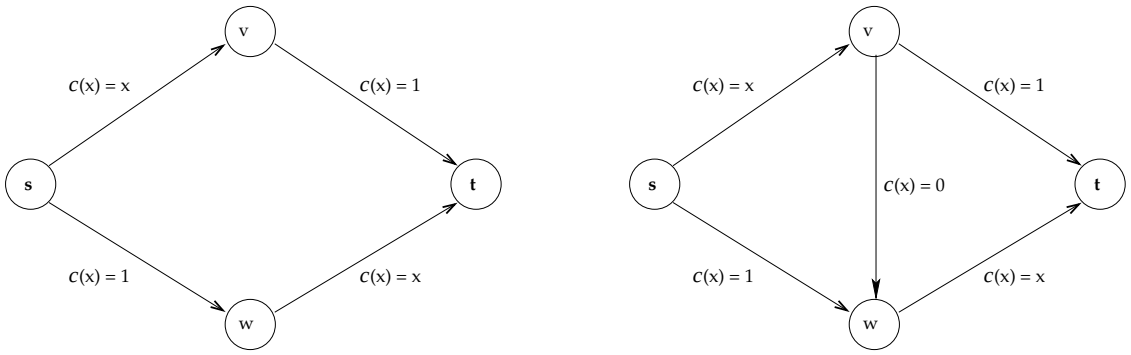
With this lecture we begin the second part of the course. In many settings, you do not have the luxury of designing a game from scratch. When do games "in the wild" have near-optimal equilibria?

Unlike all of our carefully crafted DSIC mechanisms, games in the wild generally have no dominant strategies. As a consequence, predicting the game's outcome requires stronger assumptions about how players behave. There's also no reason to expect optimal outcomes in games that we didn't design. Even for approximation results, we can only hope for success in fairly specific — though hopefully interesting and relevant — application domains.

In all of our mechanism design applications, strategic players held private information. In this part of the course, we'll focus only on full-information games, where the payoffs of all players are common knowledge. For example, when we consider network traffic, we'll assume that everyone prefers shorter routes to longer routes. By contrast, in a single-item auction with private information, one bidder has no idea whether a different bidder would prefer to win at some price p or would prefer to lose (at price 0). In later advanced lectures, we'll develop techniques for quantifying the inefficiency of equilibria also in games with incomplete information, such as auctions. The research agendas of mechanism design and quantifying inefficiency in games are not at odds, and have usefully intertwined in the past few years.

The foundations of mechanism design were laid decades ago in the economics literature; while the computer science community has helped advance the basic theory, its bigger contributions have been through novel questions, mathematical techniques, and applications. By contrast, all of the research from this part of the course will be from the last 15 (and mostly 5-10) years, and the origins of the area are in theoretical computer science [3, 6].

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.



(a) Initial network (b) Augmented network

Figure 1: Braess's Paradox. After the addition of the (v, w) edge, the price of anarchy is 4/3.

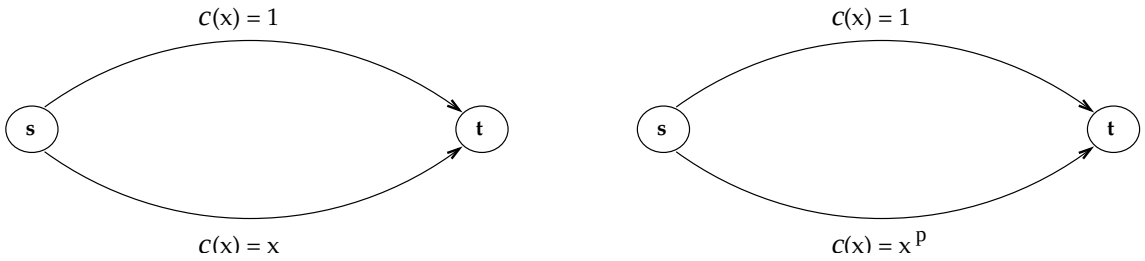
2 Selfish Routing: Examples

2.1 Braess's Paradox

Recall Braess's Paradox from Lecture 1 (Figure 1) [1]. One unit of traffic (e.g., rush-hour drivers) leave from s to t at the same time. Each edge is labeled with a cost function, which describes the travel time of all traffic on the edge, as a function of the fraction x of traffic that uses it. In the network in Figure 1(a), by symmetry, in equilibrium half of the traffic uses each route, and the travel time experienced by all traffic is $\frac{3}{2}$. After installing a teleportation device allowing drivers to travel instantly from v to w (Figure 1(b)), however, it is a dominant strategy for every driver to take the new route $s \to v \to w \to t$. The common travel time in this new equilibrium is 2. The minimum-possible travel time in the new network is $\frac{3}{2}$ — there is no profitable way to use the teleporter. If we define the *price of anarchy (POA)* to be the ratio between the travel time in an equilibrium and the minimum-possible average travel time, then the price of anarchy in the Braess network (Figure 1(b)) is $\frac{4}{3}$.

2.2 Pigou's Example

There is an even simpler selfish routing network in which the POA is $\frac{4}{3}$, first discussed in 1920 by Pigou [4]. In Pigou's example (Figure 2(a)), every driver has a dominant strategy to take the lower link — even when congested with all of the traffic, it is no worse than the alternative. Thus, in equilibrium all drivers use the lower edge and experience travel time 1. Can we do better? Sure — any other solution is better! An altruistic dictator would minimize the average travel time by splitting the traffic equally between the two links. This results in an average travel time of $\frac{3}{4}$, showing that the POA in Pigou's example is $\frac{4}{3}$.



(a) Pigou's example (b) A nonlinear variant

Figure 2: Pigou's example and a nonlinear variant. The cost function $c(x)$ describes the cost incurred by users of an edge, as a function of the amount of traffic routed on the edge.

2.3 Nonlinear Pigou's Example

The POA is $\frac{4}{3}$ in both Braess's Paradox and Pigou's example — not so bad for completely unregulated behavior. The story is not so rosy in all networks, however. In the nonlinear Pigou's example (Figure 2(b)), we replace the previous cost function $c(x) = x$ of the lower edge with the function $c(x) = x^p$, with p large. The lower edge remains a dominant strategy, and the equilibrium travel time remains 1. What's changed is that the optimal solution is now much better. If we again split the traffic equally between the two links, then the average travel time tends to $\frac{1}{2}$ as $p \to \infty$ — traffic on the bottom edge gets to t nearly instantaneously. We can do even better by routing $(1-\epsilon)$ traffic on the bottom link, where ϵ tends to 0 as p tends to infinity — almost all of the traffic gets to t with travel time $(1-\epsilon)^p$, which is close to 0 when p is sufficiently large, and the ϵ fraction of martyrs on the upper edge contribute little to the average travel time. We conclude that the POA in the nonlinear Pigou's example is unbounded as $p \to \infty$.

3 Main Result: Statement and Interpretation

The POA of selfish routing can be large (Section 2.3) or small (Section 2.1 and 2.2). The goal of this lecture is to provide a thorough understanding of when the POA of selfish routing is close to 1. Looking at our three examples thus far, we see that "highly nonlinear" cost functions can prevent a selfish routing network from having a POA close to 1, while our two examples with linear cost functions have a small POA. The coolest statement that might be true is that highly nonlinear cost functions are the *only* obstacle to a small POA — that every network with not-too-nonlinear cost functions, no matter how complex, has POA close to 1. The main result of this lecture formulates and proves this conjecture.

The model we study is the following. There is a directed graph $G = (V, E)$, with a source vertex s and a sink vertex t. There are r units of traffic (or *flow*) destined for t from s.[1] We

[1]To minimize notation, we prove the main result only for single-commodity networks, where there is one source and one sink. The main result and its proof extend easily to networks with multiple sources and

treat G as a flow network, in the sense of the maximum- and minimum-cost flow problems.

Each edge e of the network has a *cost function*, describing travel time (per unit of traffic) as a function of the amount of traffic on the edge. Edges do not have explicit capacities. For our selfish routing lectures, we always assume that every cost function is non-negative, continuous, and nondecreasing. These are very mild assumptions in most relevant applications, like road or communication network traffic.

We first state an informal version of this lecture's main result and explain how to interpret and use it. We give a formal statement at the end of this section and a proof in Section 5. Importantly, the theorem is parameterized by a set $\mathcal{C}$ of permissible cost functions. This reflects our intuition that the POA of selfish routing seems to depend on the "degree of nonlinearity" of the network's cost functions. The result is already interesting for simple classes $\mathcal{C}$ of cost functions, such as the set $\{c(x) = ax + b : a, b \geq 0\}$ of affine functions with nonnegative coefficients.

Theorem 3.1 (Tight POA Bounds for Selfish Routing (Informal) [5]) *Among all networks with cost functions in a set $\mathcal{C}$, the largest POA is achieved in a Pigou-like network.*

The point of Theorem 3.1 is that worst-case examples are always simple. The principal culprit for inefficiency in selfish routing is nonlinear cost functions, not complex network structure.

For a particular cost function class $\mathcal{C}$ of interest, Theorem 3.1 reduces the problem of computing the worst-case POA to a back-of-the-envelope calculation. Without Theorem 3.1, one would effectively have to search through all networks with cost functions in $\mathcal{C}$ to find the one with the largest POA. Theorem 3.1 guarantees that the much simpler search through Pigou-like examples is sufficient.

For example, when $\mathcal{C}$ is the set of affine cost functions, Theorem 3.1 implies that Pigou's example (Section 2.2) maximizes the POA. Thus, the POA is always at most $\frac{4}{3}$ in selfish routing networks with affine cost functions. When $\mathcal{C}$ is the set of polynomials with nonnegative coefficients and degree at most d, Theorem 3.1 implies that the worst example is the nonlinear Pigou's example (Section 2.3). It is straightforward to compute the POA in this worst-case example, and the result of this computation is an upper bound on the POA of every selfish routing network with such cost functions. See Table 1 for some examples, which makes clear the point that the POA of selfish routing is large only in networks with "highly nonlinear" cost functions. For example, quartic functions have been proposed as a reasonable model of road traffic in some situations [7]. The worst-case POA with respect to such function is slightly above 2. We discuss cost functions germane to communication networks in the next lecture.

We next work toward a formal version of Theorem 3.1. This requires formalizing the "Pigou-like networks" for a class $\mathcal{C}$ of cost functions. We then define a lower bound on the POA based solely on these trivial instances. The formal version of Theorem 3.3 states a matching upper bound on the POA of every selfish routing network with cost functions in $\mathcal{C}$.

sinks; see the Exercises.

Table 1: The worst-case POA in selfish routing networks with cost functions that are polynomials with nonnegative coefficients and degree at most d.

Description	Typical Representative	Price of Anarchy
Linear	$ax + b$	$4/3$
Quadratic	$ax^2 + bx + c$	$\frac{3\sqrt{3}}{3\sqrt{3}-2} \approx 1.6$
Cubic	$ax^3 + bx^2 + cx + d$	$\frac{4\sqrt[3]{4}}{4\sqrt[3]{4}-3} \approx 1.9$
Quartic	$ax^4 + \cdots$	$\frac{5\sqrt[4]{5}}{5\sqrt[4]{5}-4} \approx 2.2$
Polynomials of degree $\leq d$	$\sum_{i=0}^{d} a_i x^i$	$\frac{(d+1)\sqrt[d]{d+1}}{(d+1)\sqrt[d]{d+1}-d} \approx \frac{d}{\ln d}$

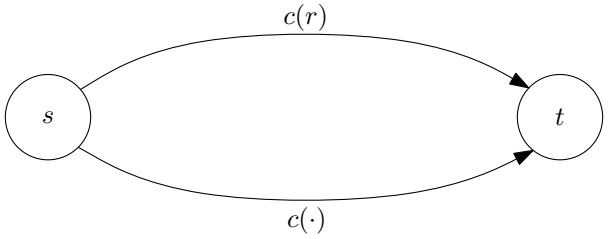


Figure 3: Pigou-like network.

Definition 3.2 (Pigou-like Network) A *Pigou-like network* has:

- Two vertices, s and t.
- Two edges from s to t.
- A traffic rate $r > 0$.
- A cost function $c(\cdot)$ on the first edge.
- The cost function everywhere equal to $c(r)$ on the second edge.

See also Figure 3. Note that there are two free parameters in the description of a Pigou-like network, the traffic rate r and the cost function $c(\cdot)$ of the first edge.

The POA of a Pigou-like network is easy to understand. By construction, the lower edge is a dominant strategy for all traffic — it is no less attractive than the alternative (with constant cost $c(r)$), even when it is fully congested. Thus, in the equilibrium all traffic travels on the lower edge, and the total travel time is $r \cdot c(r)$ — the amount of traffic times the per-unit travel time experienced by all of the traffic. We can write the minimum-possible total travel time as

$$\inf_{0 \leq x \leq r} \{x \cdot c(x) + (r - x) \cdot c(r)\}, \tag{1}$$

where x is the amount of traffic routed on the lower edge. It will be convenient later to let x range over all nonnegative reals. Since cost functions are nondecreasing, this larger range

does not change the quantity in (1) — there is always an optimal choice of x in $[0, r]$. Thus, the POA in a Pigou-like network with traffic rate $r > 0$ and upper edge cost function $c(\cdot)$ is

$$\sup_{x \geq 0} \left\{ \frac{r \cdot c(r)}{x \cdot c(x) + (r - x) \cdot c(r)} \right\}.$$

Let $\mathcal{C}$ be an arbitrary set of nonnegative, continuous, and nondecreasing cost functions. Define the *Pigou bound* $\alpha(\mathcal{C})$ as the worst POA in a Pigou-like network with cost functions in $\mathcal{C}$. Formally,

$$\alpha(\mathcal{C}) := \sup_{c \in \mathcal{C}} \sup_{r \geq 0} \sup_{x \geq 0} \left\{ \frac{r \cdot c(r)}{x \cdot c(x) + (r - x) \cdot c(r)} \right\}. \tag{2}$$

The first two suprema simply search over all choices of the two free parameters $c \in \mathcal{C}$ and $r \geq 0$ in a Pigou-like network; the third computes the best-possible routing in the chosen Pigou-like network.

The Pigou bound can be evaluated explicitly for many sets $\mathcal{C}$ of interest. For example, if $\mathcal{C}$ is the set of affine (or even concave) functions, then $\alpha(\mathcal{C}) = \frac{4}{3}$; see the Problems. The expressions in Table 1 are precisely the Pigou bounds for sets of polynomials with nonnegative coefficients and bounded degree. The Pigou bound is achieved for these sets of cost functions by the nonlinear Pigou example (Section 2.3).

By construction, if $\mathcal{C}$ contains all of the constant functions, then $\alpha(\mathcal{C})$ is a lower bound on the worst-case POA of selfish routing networks with cost functions in $\mathcal{C}$ — even for just the Pigou-like networks in this family.[2] The formal statement of Theorem 3.1 is that the Pigou bound $\alpha(\mathcal{C})$ is an upper bound on the POA of *every* selfish routing network with cost functions in $\mathcal{C}$, whether Pigou-like or not.

Theorem 3.3 (Tight POA Bounds for Selfish Routing (Formal) [5, 2]) *For every set $\mathcal{C}$ of cost functions and every selfish routing network with cost functions in $\mathcal{C}$, the POA is at most $\alpha(\mathcal{C})$.*

4 Technical Preliminaries

Before proving Theorem 3.3, we review some flow network preliminaries. While notions like flow and equilibria are easy to define in Pigou-like networks, defining them in general networks requires a little care.

Let $G = (V, E)$ be a selfish routing network, with r units of traffic traveling from s to t. Let $\mathcal{P}$ denote the set of s-t paths of G, which we assume is non-empty. A *flow* describes how traffic is split over the s-t paths — it is a non-negative vector $\{f_P\}_{P \in \mathcal{P}}$ with $\sum_{P \in \mathcal{P}} f_P = r$. For example, in Figure 4, half of the traffic takes the zig-zag path $s \to v \to w \to t$, while the other half is split equally between the two two-hop paths.

[2] As long as $\mathcal{C}$ contains at least one function c with $c(0) > 0$, the Pigou bound is a lower bound on the POA of selfish routing networks with cost functions in $\mathcal{C}$. The reason is that Pigou-like networks can be simulated by slightly more complex networks under this weaker assumption; see the Exercises for details.

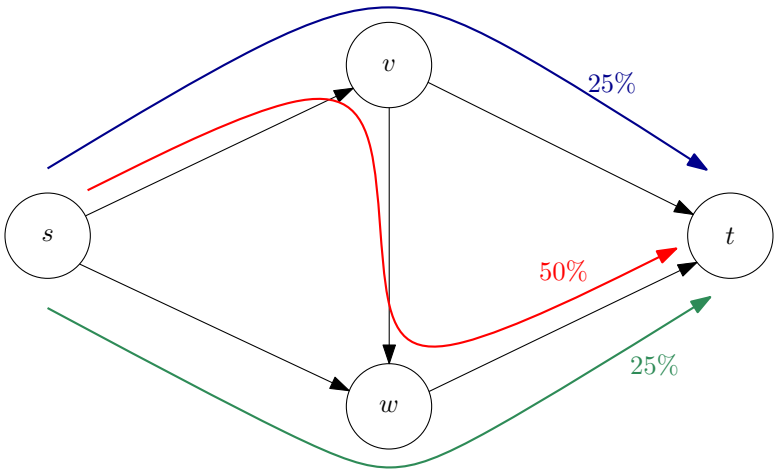


Figure 4: Example flow.

For an edge $e \in E$ and a flow f, we write $f_e = \sum_{P \in \mathcal{P}\,:\,e \in P} f_p$ for the amount of flow that uses a path that includes e. For example, in Figure 4, $f_{(s,v)} = f_{(w,t)} = \frac{3}{4}$, $f_{(s,w)} = f_{(v,t)} = \frac{1}{4}$, and $f_{(v,w)} = \frac{1}{2}$.

A flow is an *equilibrium* if and only if traffic travels only on shortest s-t paths. Formally, $f_{\widehat{P}} > 0$ only if

$$\widehat{P} \in \operatorname*{argmin}_{P \in \mathcal{P}} \left\{ \underbrace{\sum_{e \in P} c_e(f_e)}_{:= c_P(f)} \right\}.$$

Note that "shortest" is defined using the travel times $\{c_e(f_e)\}$ with respect to the flow f. For example, the flow in Figure 4 is not an equilibrium because the only shortest path is the zig-zag path, and some of the flow doesn't use it.

A non-obvious fact is that, in every selfish routing network, there is at least one equilibrium flow. We'll sketch a proof of this later, in Lecture 13.[3]

Our objective function is the total travel time incurred by traffic, and this is denoted by $C(f)$ for a flow f. We sometimes call the total travel time the *cost* of a flow. This objective function can be computed in two different ways, and both ways are useful. First, we can tally travel time path-by-path:

$$C(f) = \sum_{P \in \mathcal{P}} f_P \cdot c_P(f).$$

Alternatively, we can tally it edge-by-edge:

$$C(f) = \sum_{e \in E} f_e \cdot c_e(f_e).$$

[3]Our assumption that cost functions are continuous is important for this existence result.

Recalling that $c_P(f) = \sum_{e \in P} c_e(f_e)$ and $f_e = \sum_{P \in \mathcal{P}\,:\, e \in P} f_P$ by definition, a simple reversal of sums formally shows the equivalence of the two expressions above.

A second non-trivial fact about equilibrium flows is that all such flows have the same cost; we'll sketch a proof of this in a later lecture. Thus, it makes sense to define the *price of anarchy (POA)* of a selfish routing network as the ratio between the cost of an equilibrium flow and the cost of an optimal (minimum-cost) flow.

5 Main Result: Proof

We now prove Theorem 3.3. Fix a network G with cost functions in $\mathcal{C}$. Let f and f^* denote equilibrium and optimal (minimum-cost) flows, respectively. The proof has two parts.

The first part of the proof shows that if we "freeze" all edge costs at their equilibrium values $c_e(f_e)$, then the equilibrium flow f is in fact optimal. This should be intuitive since an equilibrium flow routes all traffic on shortest paths with respect to the edge travel times it induces. This does not contradict the sub-optimality of equilibrium flows, as other flows generally induce different edge costs.

Formally, since f is an equilibrium flow, if $f_{\widehat{P}} > 0$, then $c_{\widehat{P}}(f) \leq c_P(f)$ for all $P \in \mathcal{P}$. In particular, all paths $\widehat{P}$ used by the equilibrium flow have a common cost $c_{\widehat{P}}(f)$, call it L. Moreover, $c_P(f) \geq L$ for every path $P \in \mathcal{P}$. Thus,

$$\sum_{P \in \mathcal{P}} \underbrace{f_P}_{\text{sums to } r} \cdot \underbrace{c_P(f)}_{= L \text{ if } f_P > 0} = r \cdot L \tag{3}$$

while

$$\sum_{P \in \mathcal{P}} \underbrace{f^*_P}_{\text{sums to } r} \cdot \underbrace{c_P(f)}_{\geq L} \geq r \cdot L. \tag{4}$$

Rewriting the left-hand sides of (3) and (4) as sums over edges and subtracting (3) from (4) yields

$$\sum_{e \in E} (f^*_e - f_e) c_e(f_e) \geq 0. \tag{5}$$

The "variational inequality" in (5) is stating something very intuitive: since the equilibrium flow f routes all traffic on shortest paths, no other flow f^* can be better if we keep all edge costs fixed at $\{c_e(f_e)\}_{e \in E}$.

The second part of the proof quantifies the extent to which the optimal flow f^* can be better than f. The rough idea is to show that, edge by edge, the gap in costs between f and f^* is no worse than the Pigou bound. This statement only holds up to an error term for each edge, but we can control the sum of the error terms using the inequality (5) from the first part of the proof.

Formally, for each edge $e \in E$, instantiate the right-hand side of the Pigou bound (2) using c_e for c, f_e for r, and f^*_e for x. Since $\alpha(\mathcal{C})$ is a supremum over all possible choices of

c, r, and x, we have

$$\alpha(\mathcal{C}) \geq \frac{f_e \cdot c_e(f_e)}{f_e^* \cdot c_e(f_e^*) + (f_e - f_e^*)c_e(f_e)}.$$

Note that the definition of the Pigou bound accommodates both the cases $f_e^* \leq f_e$ and $f_e^* \geq f_e$. Rearranging,

$$f_e^* \cdot c_e(f_e^*) \geq \frac{1}{\alpha(\mathcal{C})} \cdot f_e \cdot c_e(f_e) + (f_e^* - f_e)c_e(f_e).$$

Summing this inequality over all $e \in E$ gives

$$C(f^*) \geq \frac{1}{\alpha(\mathcal{C})} \cdot C(f) + \underbrace{\sum_{e \in E}(f_e^* - f_e)c_e(f_e)}_{\geq 0 \text{ by (5)}} \geq \frac{C(f)}{\alpha(\mathcal{C})}.$$

Thus $C(f)/C(f^*) \leq \alpha(\mathcal{C})$, and the proof of Theorem 3.3 is complete.

References

[1] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[2] J. R. Correa, A. S. Schulz, and N. E. Stier Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.

[3] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1563 of *Lecture Notes in Computer Science*, pages 404–413, 1999.

[4] A. C. Pigou. *The Economics of Welfare*. Macmillan, 1920.

[5] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal of Computer and System Sciences*, 67(2):341–364, 2003.

[6] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[7] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, 1985.

CS364A: Algorithmic Game Theory
Lecture #12: More on Selfish Routing*

Tim Roughgarden†

October 30, 2013

1 Case Study: Network Over-Provisioning

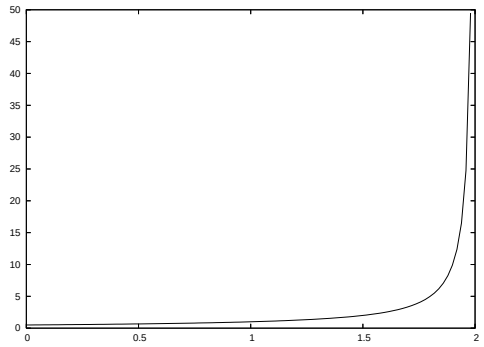
1.1 Motivation

The selfish routing model introduced last lecture can provide insight into many different kinds of networks, including transportation, communication, and electrical networks. One big advantage in communication networks is that it's often relatively cheap to add additional capacity to a network. Because of this, a popular strategy to communication network management is to install more capacity than is needed, meaning that the network will generally not be close to fully utilized (see e.g. [4]).

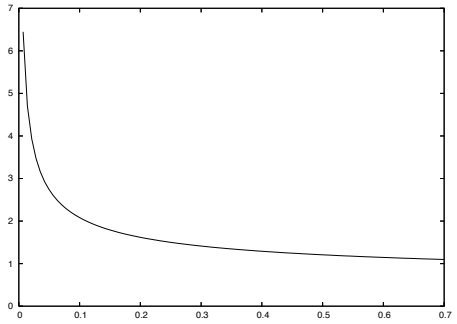
There are several reasons why network over-provisioning is common in communication networks. One reason is to anticipate future growth in demand. Beyond this, it has been observed empirically that networks tend to perform better — for example, suffering fewer packet drops and delays — when they have extra capacity. Network over-provisioning has been used as an alternative to directly enforcing "quality-of-service (QoS)" guarantees (e.g., delay bounds), for example via an admission control protocol that refuses entry to new traffic when too much congestion would result [4].

The goal of this section is develop theory to corroborate the empirical observation that network over-provisioning leads to good performance. Section 1.2 shows how to apply directly the theory developed last lecture to over-provisioned networks. Section 1.3 offers a second approach to proving the same point, that selfish routing with extra capacity is competitive with optimal routing.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.



(a) M/M/1 delay function



(b) Extra capacity vs. POA curve

Figure 1: Modest overprovisioning guarantees near-optimal routing. The left-hand figure displays the per-unit cost $c(x) = 1/(u - x)$ as a function of the load x for an edge with capacity $u = 2$. The right-hand figure shows the worst-case price of anarchy as a function of the fraction of unused network capacity.

1.2 POA Bounds for Over-Provisioned Networks

The optimal price of anarchy (POA) bounds for selfish routing developed last lecture are parameterized by the class of permissible network cost functions. In this section, we consider a network in which every cost function $c_e(x)$ has the form

$$c_e(x) = \begin{cases} \frac{1}{u_e - x} & \text{if } x < u_e \\ +\infty & \text{if } x \geq u_e. \end{cases} \tag{1}$$

The parameter u_e should be thought of as the capacity of edge e. A cost function of the form (1) is the expected delay in an M/M/1 queue, meaning a queue where jobs arrive according to a Poisson process with rate x and have independent and exponentially distributed services times with mean $1/u_e$. This is generally the first and simplest cost function used to model delays in communication networks (e.g. [2]). Figure 1(a) displays such a function; it stays very flat until the traffic load nears the capacity, at which point the cost rapidly tends to $+\infty$.

For a parameter $\beta \in (0, 1)$, call a selfish routing network with M/M/1 delay functions *β-over-provisioned* if $f_e \leq (1 - \beta)u_e$ for every edge e, where f is an equilibrium flow. That is, at equilibrium, the maximum link utilization in the network is at most $(1 - \beta) \cdot 100\%$.

Figure 1(a) suggests the following intuition: when β is not too close to 0, the equilibrium flow is not too close to the capacity on any edge, and in this range the edges' cost functions behave like low-degree polynomials with nonnegative coefficients. Last lecture we saw that the POA is small in networks with such cost functions.

More formally, the main theorem from last lecture reduces computing the worst-case POA in arbitrary β-over-provisioned selfish routing networks to computing the worst-case POA merely in β-over-provisioned Pigou-like examples. A computation, which the reader

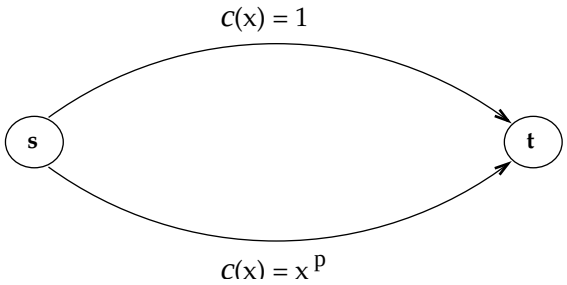


Figure 2: Nonlinear variant of Pigou's example. The POA of selfish routing can be arbitrarily large.

is encouraged to do in the privacy of their own home, shows that the worst-case POA in β-over-provisioned networks is at most

$$\frac{1}{2}\left(1+\sqrt{\frac{1}{\beta}}\right); \tag{2}$$

see Figure 1(b). As we've come to expect, very simple networks show that the bound in (2) is tight for all values of $\beta \in (0,1)$.

Unsurprisingly, the bound in (2) tends to 1 as β tends to 1 and to $+\infty$ as β tends to 0; these are the cases where the cost functions effectively act like constant functions and like very high-degree polynomials, respectively. What's interesting to investigate is intermediate values of β. For example, if $\beta = .1$ — meaning the maximum edge utilization is at most 90% — then the POA is guaranteed to be at most 2.1. In this sense, a little over-provisioning is sufficient for near-optimal selfish routing, corroborating what has been empirically observed by Internet Service Providers.

1.3 A Resource Augmentation Bound

This section proves a guarantee for selfish routing in arbitrary networks, with no extra assumptions on the cost function. What could such a guarantee look like? Recall that the nonlinear variant of Pigou's example (Figure 2) shows that the POA in selfish routing networks with arbitrary cost functions is unbounded.

In this section, we compare the performance of selfish routing to a "weaker" optimal solution that is forced to send extra traffic.[1] For example, in Figure 2, with one unit of traffic, the equilibrium flow has cost 1 while the optimal flow has near-zero cost. If the optimal flow has to route *two* units of traffic through the network, then there is nowhere to hide: the best solution continues to route $(1-\epsilon)$ units of traffic on the lower edge, with the remaining $(1+\epsilon)$ units of traffic routed on the upper edge, for a total cost exceeding that of the equilibrium flow (with one unit of traffic).

[1] Another approach, explored in the systems community [5], is to instead make extra assumptions about the network structure and the traffic rate.

This "unfair" comparison between two flows at different traffic rates has an equivalent and easier to interpret formulation as a comparison between two flows with the same traffic rate but in networks with with different cost functions. Intuitively, instead of forcing the optimal flow to route additional traffic, we allow the equilibrium flow to use a "faster" network, with each original cost function $c_e(x)$ replaced by the "faster" function $c_e(\frac{x}{2})/2$. (See the Exercises for details.) This transformation is particularly easy to interpret for M/M/1 delay functions, since if $c_e(x) = 1/(u_e - x)$, then the "faster" function is $1/(2u_e - x)$ — an edge with double the capacity. The next theorem, after this reformulation, gives a second justification for network over-provisioning: a modest technology upgrade improves performance more than implementing dictatorial control.[2]

Theorem 1.1 ([6]) *For every selfish routing network and traffic rate r, the cost of an equilibrium flow with rate r is at most the cost of an optimal flow with rate $2r$.*

Proof: Fix a network G with nonnegative, nondecreasing, and continuous cost functions, and a traffic rate r. Let f and f^* denote equilibrium and optimal (minimum-cost) flows at the traffic rates r and $2r$, respectively.

The first part of the proof reuses the trick from last lecture of using fictitious cost functions, frozen at the equilibrium costs, to get a grip on the cost of the optimal flow f^*. Recall that since f is an equilibrium flow, all paths P used by f have a common cost $c_P(f)$, call it L. Moreover, $c_P(f) \geq L$ for every path $P \in \mathcal{P}$. Thus,

$$\sum_{P \in \mathcal{P}} \underbrace{f_P}_{\text{sums to } r} \cdot \underbrace{c_P(f)}_{= L \text{ if } f_P > 0} = r \cdot L \tag{3}$$

while

$$\sum_{P \in \mathcal{P}} \underbrace{f^*_P}_{\text{sums to } 2r} \cdot \underbrace{c_P(f)}_{\geq L} \geq 2r \cdot L. \tag{4}$$

That is, with respect to the fictitious costs $\{c_e(f_e)\}$, we get a great lower bound on the cost of f^* — at least twice the cost of the equilibrium flow f — much better than what we're actually trying to prove.

The second step of the proof shows that using the fictitious costs instead of the accurate ones overestimates the cost of f^* by at most the cost of f. Specifically, we complete the proof by showing that

$$\underbrace{\sum_{e \in E} f^*_e \cdot c_e(f^*_e)}_{\text{cost of } f^*} \geq \underbrace{\sum_{e \in E} f^*_e \cdot c_e(f_e)}_{\geq 2rL} - \underbrace{\sum_{e \in E} f_e \cdot c_e(f_e)}_{= rL}. \tag{5}$$

We prove that (5) holds term-by-term, that is, we show that

$$f^*_e \cdot [c_e(f_e) - c_e(f^*_e)] \leq f_e \cdot c_e(f_e) \tag{6}$$

[2]Like last lecture, we prove the result for networks with a single source and single sink. The same proof extends, with minor extra notation, to networks with multiple sources and sinks (see the Exercises).

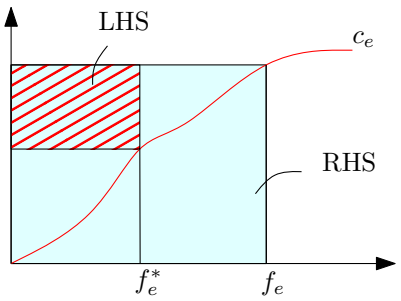


Figure 3: Proof by picture of part of Theorem 1.1.

for every edge $e \in E$. When $f_e^* \geq f_e$, the left-hand side of (6) is nonpositive and there is nothing to show. When $f_e^* < f_e$, we give a proof by picture; see Figure 3. The left-hand side of (6) is the area of the shaded region, with width f_e^* and height $c_e(f_e) - c_e(f_e^*)$. The right-hand side of (6) is the area of the solid region, with width f_e and height $c_e(f_e)$. Since $f_e^* < f_e$ and c_e is nondecreasing, the former region is a subset of the latter. This verifies (6) and completes the proof. ■

In the sense of Theorem 1.1 and its reformulation given in the Exercises, speeding up (i.e., overprovisioning) a selfish routing network by a modest amount is better than routing traffic optimally.

2 Atomic Selfish Routing

So far we've studied a *nonatomic* model of selfish routing, meaning that all players were assumed to have negligible size. This is a good model for cars on a highway or small users of a communication network, but not if a single strategic player represents, for example, all of the traffic controlled by a single Internet Service Provider. This section studies *atomic* selfish routing networks, where each player controls a non-negligible amount of traffic. While most aspects of the model will be familiar, it presents a couple of new technical complications. These complications will also be present in other classes of games that we study later.

An atomic selfish routing network has a finite number k of players. Player i has a source vertex s_i and a destination vertex t_i; these can be shared across players, or not. Each player routes 1 unit of traffic on a single s_i-t_i path, and seeks to minimize its cost.[3] Flows, equilibrium flows, and the cost of a flow are defined analogously to last lecture.

To get a feel for the atomic model, consider the variant of Pigou's example shown in Figure 4. Suppose there are two players, and recall that each controls 1 unit of flow. The optimal solution routes one player on each link, for a total cost of $1 + 2 = 3$. This is also an equilibrium flow, in the sense that neither player can decrease its cost via a unilateral

[3]Two obvious variants of the model allow players to have different sizes and/or to split traffic fractionally over multiple paths. Both variants have been extensively studied using methods similar to the ones covered in these lectures.

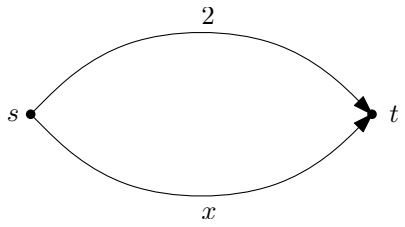


Figure 4: A pigou-like network for atomic selfish routing.

deviation. The player on the lower edge does not want to switch, since its cost would jump from 1 to 2. More interestingly, the player on the upper edge (with cost 2) has no incentive to switch to the bottom edge, where its sudden appearance would drive the cost up to 2.

There is also a *second* equilibrium in the network: if both players take the lower edge, both have a cost of 2 and neither can decrease its cost by switching to the upper edge. This equilibrium has cost 4. This illustrates an importance difference between the nonatomic and atomic models: different equilibria are guaranteed to have the same cost in the nonatomic model, but not in the atomic model.

Our current working definition of the POA — the ratio between the objective function value of an equilibrium and that of an optimal outcome — is not well defined when different equilibria have different objective function values. We extend the definition by taking a worst-case approach: the *price of anarchy (POA)* of an atomic selfish routing network is

$$\frac{\text{cost of worst equilibrium}}{\text{cost of optimal outcome}}.$$

For example, in the network in Figure 4, the POA is $\frac{4}{3}$.

A second difference between the two models is that the POA in atomic selfish routing networks can be larger than in their nonatomic counterparts. To see this, consider the four-player bidirected triangle network shown in Figure 5. Each player has two strategies, a one-hop path and a two-hop path. In the optimal flow, all players route on their one-hop paths, and the cost of this flow is 4 — these one-hop paths are precisely the four edges with the cost function $c(x) = x$. This flow is also an equilibrium flow. On the other hand, if all players route on their two-hop paths, then we obtain a second equilibrium flow. Since the first two players each incur three units of cost and the last two players each incur two units of cost, this flow has a cost of 10. As the reader should check, it is also an equilibrium. The price of anarchy of this instance is therefore $10/4 = 2.5$.

There are no worse examples with affine cost functions.[4]

Theorem 2.1 (POA Bound for Atomic Selfish Routing, Affine Cost Functions [1, 3]) *In every atomic selfish routing network with affine cost functions, the POA is at most* $\frac{5}{2}$.

[4]There are also very general and tight POA bounds known for arbitrary sets of cost functions. For example, in atomic selfish routing networks with cost functions that are polynomials with nonnegative coefficients, the POA is at most a constant that depends on the maximum polynomial degree d. The dependence on d is exponential, however, unlike the $\approx \frac{d}{\ln d}$ dependence in nonatomic selfish routing networks.

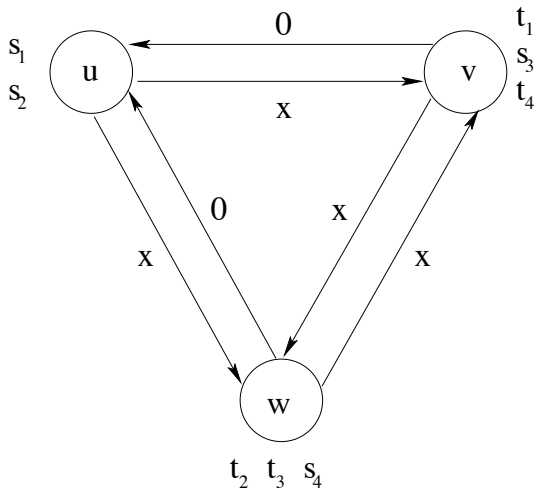


Figure 5: In atomic instances with affine cost functions, the POA can be as large as 5/2.

Proof: The following proof is a "canonical POA proof," in a sense that we'll make precise in Lecture 14. Let's just follow our nose. We need to prove a bound for every equilibrium flow; fix one f arbitrarily. Let f^* denote an optimal (minimum-cost) flow. Write f_e and f_e^* for the number of players in f and f^*, respectively, that pick a path that includes the edge e. Write each affine cost function as $c_e(x) = a_e x + b_e$ for $a_e, b_e \geq 0$.

The first step of the proof is to figure out a good way of applying our hypothesis that f is an equilibrium flow — that no player can decrease its cost through a unilateral deviation. After all, the bound of 2.5 does not generally apply to non-equilibrium flows. If we consider any player i, using path P_i in f, and any unilateral deviation to a different path $\hat{P}_i$, then we can conclude that i's equilibrium cost using P_i is at most what its cost would be if it switched to $\hat{P}_i$. This looks promising: we want an upper bound on the total cost of players in the equilibrium f, and hypothetical deviations give us upper bounds on the equilibrium costs of individual players. Which hypothetical deviations should we single out for the proof? A natural idea is to let the optimal flow f^* suggest deviations.

Formally, suppose player i uses path P_i in f and path P_i^* in f^*. Since f is an equilibrium, i's cost only increases if it switches to P_i^* (holding all other players fixed):

$$\sum_{e \in P_i} c_e(f_e) \leq \sum_{e \in P_i^* \cap P_i} c_e(f_e) + \sum_{e \in P_i^* \setminus P_i} c_e(f_e + 1),$$

where in the final term we account for the additional unit of load that i contributes to edges that it newly uses (in P_i^* but not in P_i). For all we know P_i and P_i^* are disjoint; since cost functions are nondecreasing, we have

$$\sum_{e \in P_i} c_e(f_e) \leq \sum_{e \in P_i^*} c_e(f_e + 1). \tag{7}$$

This completes the first step, in which we apply the equilibrium hypothesis to generate an upper bound (7) on the equilibrium cost of each player.

The second step of the proof sums the upper bound (7) on individual equilibrium costs over all players to get a bound on the total equilibrium cost:

$$\begin{aligned}\sum_{i=1}^{k}\sum_{e\in P_i} c_e(f_e) &\leq \sum_{i=1}^{k}\sum_{e\in P_i^*} c_e(f_e+1)\\ &= \sum_{e\in E} f_e^*\cdot c_e(f_e+1) \qquad (8)\\ &= \sum_{e\in E}\left[a_e f_e^*(f_e+1)+b_e f_e^*\right], \qquad (9)\end{aligned}$$

where in (8) we use that the term $c_e(f_e+1)$ is contributed exactly once by each player i that contemplates switching to a path P_i^* that includes the edge e — f_e^* times in all. This complete the second step of the proof.

The previous step gave an upper bound on a quantity that we care about — the cost of the equilibrium flow f — in terms of a quantity that we don't care about, the "entangled" version of f and f^* on the right-hand side of (9). The third and most technically challenging step of the proof is to "disentangle" the right-hand side of (9) and relate it to the only quantities that we do care about for a POA bound, the costs of f and f^*.

We next claim that, for every $y, z \in \{0,1,2,\ldots,\}$,

$$y(z+1) \leq \frac{5}{3}y^2 + \frac{1}{3}z^2. \qquad (10)$$

This inequality is easy to check once guessed, and we leave the verification of it as an exercise. One can check all cases where y and z are both small, and then observe that it continues to hold when either one grows large. Note that the inequality holds with equality when $y=z=1$ and when $y=1$ and $z=2$. We'll demystify how inequalities like (10) arise next week.

We now apply inequality (10) once per edge in the right-hand side of (9), with $y=f_e^*$ and $z=f_e$. We then have

$$\begin{aligned}C(f) &\leq \sum_{e\in E}\left[a_e\left(\frac{5}{3}(f_e^*)^2+\frac{1}{3}f_e^2\right)+b_e f_e^*\right]\\ &\leq \frac{5}{3}\left[\sum_{e\in E} f_e^*(a_e f_e^*+b_e)\right]+\frac{1}{3}\sum_{e\in E}a_e f_e^2\\ &\leq \frac{5}{3}\cdot C(f^*)+\frac{1}{3}\cdot C(f).\end{aligned}$$

Subtracting $\frac{1}{3}C(f)$ from both sides and multiplying through by $\frac{3}{2}$ gives

$$C(f) \leq \frac{5}{3}\cdot\frac{3}{2}\cdot C(f^*) = \frac{5}{2}\cdot C(f^*),$$

completing the proof. ■

References

[1] B. Awerbuch, Y. Azar, and L. Epstein. The price of routing unsplittable flow. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 57–66, 2005.

[2] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice-Hall, 1987. Second Edition, 1991.

[3] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–73, 2005.

[4] N. Olifer and V. Olifer. *Computer Networks: Principles, Technologies and Protocols for Network Design*. Wiley, 2005.

[5] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in Internet-like environments. *IEEE/ACM Transactions on Networking*, 14(4):725–738, 2006.

[6] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

CS364A: Algorithmic Game Theory
Lecture #13: Potential Games; A Hierarchy of Equilibria*

Tim Roughgarden†

November 4, 2013

Last lecture we proved that every pure Nash equilibrium of an atomic selfish routing game with affine cost functions (of the form $c_e(x) = a_e x + b_e$ with $a_e, b_e \geq 0$) has cost at most $\frac{5}{2}$ times that of an optimal outcome, and that this bound is tight in the worst case. There can be multiple pure Nash equilibria in such a game, and this bound of $\frac{5}{2}$ applies to all of them. But how do we know that there is at least one? After all, there are plenty of games like Rock-Paper-Scissors that possess no pure Nash equilibrium. How do we know that our price of anarchy (POA) guarantee is not vacuous? This lecture introduces basic definitions of and questions about several equilibrium concepts. When do they exist? When is computing one computationally tractable? Why should we prefer one equilibrium concept over another?

1 Potential Games and the Existence of Pure Nash Equilibria

Atomic selfish routing games are a remarkable class, in that pure Nash equilibria are guaranteed to exist.

Theorem 1.1 (Rosenthal's Theorem [4]) *Every atomic selfish routing game, with arbitrary real-valued cost functions, has at least one equilibrium flow.*

Proof: We show that every atomic selfish routing game is a *potential game*. Intuitively, we show that players are inadvertently and collectively striving to optimize a "potential function." This is one of the only general tools for guaranteeing the existence of pure Nash equilibria in a class of games.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

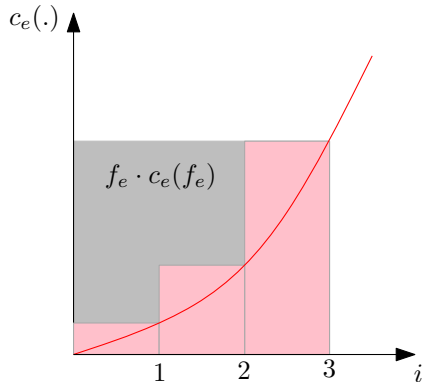


Figure 1: The function c_e and its corresponding (underestimating) potential function.

Formally, define a *potential function* on the set of flows of an atomic selfish routing game by

$$\Phi(f) = \sum_{e \in E} \sum_{i=1}^{f_e} c_e(i), \tag{1}$$

where f_e is the number of player that choose a path in f that includes the edge e. The inner sum in (1) is the "area under the curve" of the cost function c_e; see Figure 1. Contrast this with the corresponding term $f_e \cdot c_e(f_e)$ in the cost objective function we studied last week, which corresponds to the shaded bounding box in Figure 1. The similarity between the potential function and the cost objective function can be useful, as we'll see later.

The defining condition of a potential function is the following. Consider an arbitrary flow f, and an arbitrary player i, using the s_i-t_i path P_i in f, and an arbitrary deviation to some other s_i-t_i path $\hat{P}_i$. Let $\hat{f}$ denote the flow after i's deviation from P_i to $\hat{P}_i$. Then,

$$\Phi(\hat{f}) - \Phi(f) = \sum_{e \in \hat{P}_i} c_e(\hat{f}_e) - \sum_{e \in P_i} c_e(f_e); \tag{2}$$

that is, the change in the potential function under a unilateral deviation is exactly the same as the change in the deviator's individual cost. In this sense, the single function Φ simultaneously tracks the effect of deviations by each of the players.

Once the potential function Φ is correctly guessed, the property (2) is easy to verify. Looking at (1), we see that the inner sum of the potential function corresponding to edge e picks up an extra term $c_e(f_e + 1)$ whenever e is newly used by player i (i.e., in $\hat{P}_i$ but not P_i), and sheds its final term $c_e(f_e)$ when e is newly unused by player i. Thus, the left-hand side of (2) is

$$\sum_{e \in \hat{P}_i \setminus P_i} c_e(f_e + 1) - \sum_{P_i \setminus \hat{P}_i} c_e(f_e),$$

which is exactly the same as the right-hand side of (2).

Given the potential function Φ, the proof of Theorem 1.1 is easy. Let f denote the flow that minimizes Φ — since there are only finitely many flows, such a flow exists. Then, no

unilateral deviation by any player can decrease Φ. By (2), no player can decrease its cost by a unilateral deviation and so f is an equilibrium flow. ■

2 Extensions

The proof idea in Theorem 1.1 can be used to prove a number of other results. First, the proof remains valid for arbitrary cost functions, nondecreasing or otherwise. We'll use this fact in Lecture 15, when we discuss a class of games with "positive externalities."

Second, the proof of Theorem 1.1 never uses the network structure of a selfish routing game. That is, the argument remains valid for *congestion games*, the generalization of atomic selfish routing games in which there is an abstract set E of resources (previously, edges), each with a cost function, and each player i has an arbitrary collection $S_i \subseteq 2^E$ of strategies (previously, s_i-t_i paths), each a subset of resources. We'll discuss congestion games at length in Lecture 19.

Finally, analogous arguments apply to the *nonatomic* selfish routing networks introduced in Lecture 11. We sketch the arguments here; details are in [5]. Since players have negligible size in such games, we replace the inner sum in (1) by an integral:

$$\Phi(f) = \sum_{e \in E} \int_0^{f_e} c_e(x)dx, \tag{3}$$

where f_e is the amount of traffic routed on edge e by the flow f. Because cost functions are assumed continuous and nondecreasing, the function Φ is continuously differentiable and convex. The first-order optimality conditions of Φ are precisely the equilibrium conditions of a flow in a nonatomic selfish routing network (see Exercises). This gives a sense in which the local minima of Φ correspond to equilibrium flows. Since Φ is continuous and the space of all flows is compact, Φ has a global minimum, and this flow must be an equilibrium. This proves existence of equilibrium flows in nonatomic selfish routing networks. Uniqueness of such flows follows from the convexity of Φ — its only local minima are its global minima. When Φ has multiple global minima, all with the same potential function value, these correspond to multiple equilibrium flows that all have the same total cost.

3 A Hierarchy of Equilibrium Concepts

How should we discuss the POA of a game with no pure Nash equilibria (PNE)? In addition to games like Rock-Paper-Scissors, atomic selfish routing games with varying player sizes need not have PNE, even with only two players and quadratic cost functions [5, Example 18.4]. For a meaningful POA analysis of such games, we need to enlarge the set of equilibria to recover guaranteed existence. The rest of this lecture introduces three relaxations of PNE, each more permissive and more computationally tractable than the previous one (Figure 2). All three of these more permissive equilibrium concepts are guaranteed to exist in every finite game.

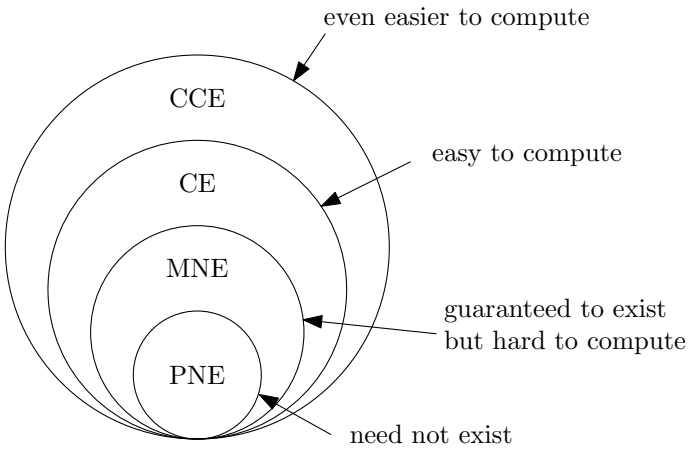


Figure 2: The Venn-diagram of the hierarchy of equilibrium concepts.

3.1 Cost-Minimization Games

A *cost-minimization* game has the following ingredients:

- a finite number k of players;
- a finite strategy set S_i for each player i;
- a cost function $C_i(\mathbf{s})$ for each player i, where $\mathbf{s} \in S_1 \times \cdots \times S_k$ denotes a *strategy profile* or *outcome*.

For example, atomic routing games are cost-minimization games, with $C_i(\mathbf{s})$ denoting i's travel time on its chosen path, given the paths $\mathbf{s}_{-i}$ chosen by the other players.

Conventionally, the following equilibrium concepts are defined for payoff-maximization games, with all of the inequalities reversed. The two definitions are completely equivalent.

3.2 Pure Nash Equilibria (PNE)

Recall the definition of a PNE: unilateral deviations can only increase a player's cost.

Definition 3.1 A strategy profile $\mathbf{s}$ of a cost-minimization game is a *pure Nash equilibrium (PNE)* if for every player $i \in \{1, 2, \ldots, k\}$ and every unilateral deviation $s_i' \in S_i$,

$$C_i(\mathbf{s}) \leq C_i(s_i', \mathbf{s}_{-i}). \tag{4}$$

PNE are easy to interpret but, as discussed above, do not exist in all games of interest. We leave the POA of pure Nash equilibria undefined in games without at least one PNE.

3.3 Mixed Nash Equilibria (MNE)

When we discussed the Rock-Paper-Scissors game in Lecture 1, we introduced the idea of a player randomizing over its strategies via a *mixed strategy*. In a mixed Nash equilibrium, players randomize independently and unilateral deviations can only increase a player's expected cost.

Definition 3.2 Distributions $\sigma_1, \ldots, \sigma_k$ over strategy sets $S_1, \ldots, S_k$ of a cost-minimization game constitute a *mixed Nash equilibrium (MNE)* if for every player $i \in \{1, 2, \ldots, k\}$ and every unilateral deviation $s_i' \in S_i$,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i', \mathbf{s}_{-i})]\,, \tag{5}$$

where σ denotes the product distribution $\sigma_1 \times \cdots \times \sigma_k$.

Definition 3.2 only considers pure-strategy unilateral deviations; also allowing mixed-strategy unilateral deviations does not change the definition (Exercise).

By the definitions, every PNE is the special case of MNE in which each player plays deterministically. The Rock-Paper-Scissors game shows that, in general, a game can have MNE that are not PNE.

Here are two highly non-obvious facts that we'll discuss at length in Lecture 20. First, every cost-minimization game has at least one MNE; this is Nash's theorem [3]. Second, computing a MNE appears to be a computationally intractable problem, even when there are only two players. For now, by "seems intractable" you can think of as being roughly NP-complete; the real story is more complicated, as we'll discuss in the last week of the course.

The guaranteed existence of MNE implies that the POA of MNE is well defined in every finite game — this is an improvement over PNE. The computational intractability of MNE raises the concern that POA bounds for them need not be meaningful. If we don't expect the players of a game to quickly reach an equilibrium, why should we care about performance guarantees for equilibria? This objection motivates the search for still more permissive and computationally tractable equilibrium concepts.

3.4 Correlated Equilibria (CE)

Our next equilibrium notion takes some getting used to. We define it, then explain the standard semantics, and then offer an example.

Definition 3.3 A distribution σ on the set $S_1 \times \cdots \times S_k$ of outcomes of a cost-minimization game is a *correlated equilibrium (CE)* if for every player $i \in \{1, 2, \ldots, k\}$, strategy $s_i \in S_i$, and every deviation $s_i' \in S_i$,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s}) \mid s_i] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i', \mathbf{s}_{-i}) \mid s_i]\,. \tag{6}$$

Importantly, the distribution σ in Definition 3.3 need not be a product distribution; in this sense, the strategies chosen by the players are correlated. Indeed, the MNE of a game correspond to the CE that are product distributions (see Exercises). Since MNE are guaranteed to exist, so are CE. Correlated equilibria also have a useful equivalent definition in terms of "switching functions;" see the Exercises.

The usual interpretation of a correlated equilibrium [1] involves a trusted third party. The distribution σ over outcomes is publicly known. The trusted third party samples an outcome $\mathbf{s}$ according to σ. For each player $i = 1, 2, \ldots, k$, the trusted third party privately suggests the strategy s_i to i. The player i can follow the suggestion s_i, or not. At the time of decision-making, a player i knows the distribution σ, one component s_i of the realization σ, and accordingly has a posterior distribution on others' suggested strategies $\mathbf{s}_{-i}$. With these semantics, the correlated equilibrium condition (6) requires that every player minimizes its expected cost by playing the suggested strategy s_i. The expectation is conditioned on i's information — σ and s_i — and assumes that other players play their recommended strategies $\mathbf{s}_{-i}$.

Believe it or not, a traffic light is a perfect example of a CE that is not a MNE. Consider the following two-player game:

	stop	go
stop	0,0	0,1
go	1,0	-5,-5

If the other player is stopping at an intersection, then you would rather go and get on with it. The worst-case scenario, of course, is that both players go at the same time and get into an accident. There are two PNE, (stop,go) and (go,stop). Define σ by randomizing 50/50 between these two PNE. This is not a product distribution, so it cannot correspond to a MNE of the game. It is, however, a CE. For example, consider the row player. If the trusted third party (i.e., the stoplight) recommends the strategy "go" (i.e., is green), then the row player knows that the column player was recommended "stop" (i.e., has a red light). Assuming the column player plays its recommended strategy (i.e., stops at the red light), the best response of the row player is to follow its recommendation (i.e., to go). Similarly, when the row player is told to stop, it assumes that the column player will go, and under this assumption stopping is a best response.

In Lecture 18 we'll prove that, unlike MNE, CE are computationally tractable. One proof goes through linear programming. More interesting, and the focus of our lectures, is the fact that there are distributed learning algorithms that quickly guide the history of joint play to the set of CE.

3.5 Coarse Correlated Equilibria (CCE)

We should already be quite pleased with positive results, like good POA bounds, that apply to the computationally tractable set of CE. But if we can get away with it, we'd be happy to enlarge the set of equilibria even further, to an "even more tractable" concept.

Definition 3.4 ([2]) A distribution σ on the set $S_1 \times \cdots \times S_k$ of outcomes of a cost-minimization game is a *coarse correlated equilibrium (CCE)* if for every player $i \in \{1, 2, \ldots, k\}$ and every unilateral deviation $s_i' \in S_i$,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i', \mathbf{s}_{-i})]\,. \tag{7}$$

In the equilibrium condition (7), when a player i contemplates a deviation s_i', it knows only the distribution σ and *not* the component s_i of the realization. Put differently, a CCE only protects against unconditional unilateral deviations, as opposed to the conditional unilateral deviations addressed in Definition 3.3. Every CE is a CCE — see the Exercises — so CCE are guaranteed to exist in every finite game and are computationally tractable. As we'll see in a couple of weeks, the distributed learning algorithms that quickly guide the history of joint play to the set of CCE are even simpler and more natural than those for the set of CE.

3.6 An Example

We next consider a concrete example, to increase intuition for the four equilibrium concepts in Figure 2 and to show that all of the inclusions can be strict.

Consider an atomic selfish routing game (Lecture 12) with four players. The network is simply a common source vertex s, a common sink vertex t, and 6 parallel s-t edges $E = \{0, 1, 2, 3, 4, 5\}$. Each edge has the cost function $c(x) = x$.

The pure Nash equilibria of this game are the $\binom{6}{4}$ outcomes in which each player chooses a distinct edge. Every player suffers only unit cost in such an equilibrium. One mixed Nash equilibrium that is obviously not pure has each player independently choosing an edge uniformly at random. Every player suffers expected cost 3/2 in this equilibrium. The uniform distribution over all outcomes in which there is one edge with two players and two edges with one player each is a (non-product) correlated equilibrium, since both sides of (6) read $\frac{3}{2}$ for every i, s_i, and s_i' (see Exercises). The uniform distribution over the subset of these outcomes in which the set of chosen edges is either $\{0, 2, 4\}$ or $\{1, 3, 5\}$ is a coarse correlated equilibrium, since both sides of (7) read $\frac{3}{2}$ for every i and s_i'. It is not a correlated equilibrium, since a player i that is recommended the edge s_i can reduce its conditional expected cost to 1 by choosing the deviation s_i' to the successive edge (modulo 6).

3.7 Looking Ahead: POA Bounds for Tractable Equilibrium Concepts

The benefit of enlarging the set of equilibria is increased tractability and plausibility. The downside is that, in general, fewer desirable properties will hold.

For example, consider POA bounds, which by definition compare the objective function value of the *worst* equilibrium of a game to that of an optimal solution. The larger the set of equilibria, the worse (i.e., further from 1) the POA. Is there a "sweet spot" equilibrium concept that is simultaneously big enough to enjoy tractability and small enough to permit strong worst-case guarantees? In the next lecture, we give an affirmative answer for many interesting classes of games.

References

[1] R. J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

[2] H. Moulin and J. P. Vial. Strategically zero-sum games: The class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7(3/4):201–221, 1978.

[3] J. F. Nash. Equilibrium points in N-person games. *Proceedings of the National Academy of Science*, 36(1):48–49, 1950.

[4] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[5] T. Roughgarden. Routing games. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 18, pages 461–486. Cambridge University Press, 2007.

CS364A: Algorithmic Game Theory
Lecture #14: Robust Price-of-Anarchy Bounds in Smooth Games*

Tim Roughgarden†

November 6, 2013

1 Canonical POA Proofs

In Lecture 12 we proved that the price of anarchy (POA) in every atomic selfish routing game with affine cost functions is at most $\frac{5}{2}$. To review, the proof had the following high-level steps.

1. Given an arbitrary pure Nash equilibrium (PNE) $\mathbf{s}$, the PNE hypothesis is invoked once per player i with the hypothetical deviation s_i^*, where $\mathbf{s}^*$ is an optimal outcome, to derive the inequality $C_i(\mathbf{s}) \leq C_i(s_i^*, \mathbf{s}_{-i})$ for each i. Importantly, this is the only time that the PNE hypothesis is invoked in the entire proof.

2. The k inequalities that bound individuals' equilibrium costs are summed over the players. The left-hand side of the resulting inequality is the cost of the PNE $\mathbf{s}$; the right-hand side is a strange entangled function of $\mathbf{s}$ and $\mathbf{s}^*$ (involving terms of the form $f_e f_e^*$).

3. The hardest step is to relate the entangled term $\sum_{i=1}^{k} C_i(s_i^*, \mathbf{s}_{-i})$ generated by the previous step to the only two quantities that we care about, the costs of $\mathbf{s}$ and $\mathbf{s}^*$. Specifically, we proved an upper bound of $\frac{5}{3}\text{cost}(\mathbf{s}^*) + \frac{1}{3}\text{cost}(\mathbf{s})$. Importantly, this step is just algebra, and is agnostic to our choices of $\mathbf{s}$ and $\mathbf{s}^*$ as a PNE and an optimal outcome, respectively.

4. Solve for the POA. Subtracting $\frac{1}{3} \cdot \text{cost}(\mathbf{s})$ from both sides and multiplying through by $\frac{3}{2}$ proves that the POA is at most $\frac{5}{2}$.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

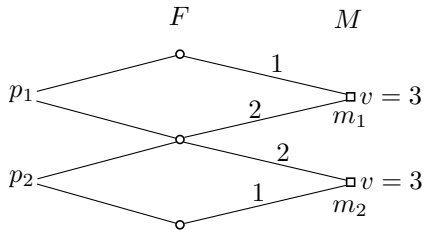


Figure 1: Location game with 3 locations (F), 2 markets (M) and 2 players.

This proof is canonical, in a sense that we formalize in this lecture. Many other POA proofs are canonical in the same sense. The main point of this lecture is that POA proofs that follow this general recipe automatically generate "robust POA bounds." That is, the proved guarantee applies not only to all PNE of the game, but also to, among other things, all of its coarse correlated equilibria — the biggest set of equilibria defined in the last lecture.

2 A Location Game

Before proceedings to the general theory, it will be helpful to have another concrete example under our belt. Consider a *location game* with the following ingredients:

- A set F of possible locations. These could represent possible locations to build a Web cache in a network, an artisinal chocolate shop in a gentrifying neighborhood, and so on.
- A set of k players. Each player i chooses one location from a set $F_i \subseteq F$ from which to provide a service. All players provide the same service; they differ only in where they are located. There is no limit on the number of markets that a player can provide service to.
- A set M of markets. Each market $j \in M$ has a value v_j that is known to all players. This is the market's maximum willingness-to-pay for receiving a service.
- For each location $\ell \in F$ and market $j \in M$, there is a known cost $c_{\ell j}$ of serving j from ℓ. This could represent physical distance, the degree of incompatibility between two technologies, and so on.

Given a strategy profile — a location choice by each player — each player tries to capture as many markets as possible, at the highest prices possible. To define the payoffs precisely, we start with an example. Figure 1 shows a location game with $F = \{1, 2, 3\}$ and $M = \{1, 2\}$. There are two players, with $F_1 = \{1, 2\}$ and $F_2 = \{2, 3\}$. Both markets have value 3. The cost between location 2 and either market is 2; locations 1 and 3 have cost 1 to the nearer market (1 and 2, respectively) and infinite cost to the other market.

Continuing the example, suppose the first player chooses location 1 and the second player chooses location 3. Then, each player has a monopoly in the market that they entered. The

only thing restricting the price charged is the maximum willingness-to-pay of the market. Thus, each player can charge 3 for its service to its market. Since the cost of service is 1 in both cases, both players have a payoff of $3 - 1 = 2$.

Alternatively, suppose the first player switches to location 2, while the second player remains at location 3. Player 1 still has a monopoly in market 1, and thus can still charge 3. Its service cost has jumped to 2, however, so its payoff from that market has dropped to 1. In market 2, player 2 can no longer charge a price of 3 without consequence — at any price strictly bigger than 2, the player 1 can profitably undercut the price and take the market. Thus, player 2 will charge the highest price it can get away with, which is 2. Since its cost of serving the market is 1, player 2's payoff is $2 - 1 = 1$.

In general, in a strategy profile $\mathbf{s}$ of a location game, the payoff of player i is defined as

$$\pi_i(\mathbf{s}) = \sum_{j \in M} \pi_{ij}(\mathbf{s}),$$

where, assuming that C is the set of chosen locations and i chooses $\ell \in C$,

$$\pi_{ij}(\mathbf{s}) = \begin{cases} 0 & \text{if } c_{\ell j} \geq v_j \text{ or } \ell \text{ is not the closest location of } C \text{ to } j \\ d_j^{(2)}(\mathbf{s}) - c_{\ell j} & \text{otherwise,} \end{cases} \tag{1}$$

where $d_j^{(2)}(\mathbf{s})$ is the highest price that player i can get away with, namely the minimum of v_j and the second-smallest cost between a location of C and j.[1]

The payoff $\pi_{ij}(\mathbf{s})$ is thus the "competitive advantage" that i has over the other players for market j, up to a cap of v_j minus the service cost. The definition in (1) assumes that each market is served by the potential provider with the lowest service cost, at the highest competitive price. This assumption can also be justified from first principles by setting up a three-stage game and proving that its "subgame perfect equilibria" have these properties; see [2] for more details.

The objective function in a location game is to maximize the social surplus. The surplus $V(\mathbf{s})$ of a strategy profile $\mathbf{s}$ — a location choice by each player — is defined as

$$V(\mathbf{s}) = \sum_{j \in M} v_j - d_j(\mathbf{s}), \tag{2}$$

where $d_j(\mathbf{s})$ is the minimum of v_j and the smallest cost between a chosen location and j. The definition (2) assumes that each market j is served by the chosen location with the smallest cost of serving j, or not at all if this cost is at least v_j.

Note that the surplus $V(\mathbf{s})$ depends on the strategy profile $\mathbf{s}$ only through the set of locations chosen by some player in $\mathbf{s}$. Indeed, the definition (2) makes sense more generally for any subset of chosen locations T, and we sometimes write $V(T)$ for this quantity.

The rest of this section proves that every PNE of every location game has social surplus at least 50% times that of an optimal outcome.

[1] In contrast to selfish routing games, these location games are most naturally described as payoff-maximization games. POA bounds are equally interesting in both formalisms. The POA of a payoff-maximization game is at most 1, the closer to 1 the better.

Theorem 2.1 ([3]) *The POA of every location game is at least* $\frac{1}{2}$.

We next identify three properties possessed by every location game; these are the only properties that our proof of Theorem 2.1 relies on.[2]

(P1) For every strategy profile $\mathbf{s}$, the sum $\sum_{i=1}^{k} \pi_i(\mathbf{s})$ of players' payoffs (i.e., the net revenue) is at most the social surplus $V(\mathbf{s})$.

This follows from the fact that each $j \in M$ contributes $v_j - d_j(\mathbf{s})$ to the surplus and $d_j^{(2)}(\mathbf{s}) - d_j(\mathbf{s})$ to the payoff of the closest location, and that $d_j^{(2)}(\mathbf{s}) \leq v_j$ by definition.

(P2) For every strategy profile $\mathbf{s}$, $\pi_i(\mathbf{s}) = V(\mathbf{s}) - V(\mathbf{s}_{-i})$. That is, a player's payoff is exactly the extra surplus created by the presence of its location.[3]

To see this property, observe that the contribution of a particular market j to the right-hand side $V(\mathbf{s})-V(\mathbf{s}_{-i})$ is the extent to which the closest chosen location to j is closer in $\mathbf{s}$ than in $\mathbf{s}_{-i}$ (with the usual upper bound v_j), namely $\min\{v_j, d_j(\mathbf{s}_{-i})\}-\min\{v_j, d_j(\mathbf{s})\}$. This is zero unless player i's location is the closest to j in $\mathbf{s}$, in which case it is

$$\min\{v_j, d_j^{(2)}(\mathbf{s})\} - \min\{v_j, d_j(\mathbf{s})\}. \tag{3}$$

Either way, this is precisely market j's contribution $\pi_{ij}(\mathbf{s})$ to player i's payoff in $\mathbf{s}$. Summing over all $j \in M$ proves the property.

(P3) The function $V(\cdot)$ is *monotone* and *submodular*, as a function of the set of chosen locations. Monotonicity just means that $V(T_1) \leq V(T_2)$ whenever $T_1 \subseteq T_2$; this property is evident from (2). Submodularity is a set-theoretic version of diminishing returns, defined formally as

$$V(T_2 \cup \{\ell\}) - V(T_2) \leq V(T_1 \cup \{\ell\}) - V(T_1) \tag{4}$$

for every location ℓ and subsets $T_1 \subseteq T_2$ of locations (Figure 2).

Submodularity follows immediately from our expression (3) for the surplus increase caused by one new location ℓ — the only interesting case is when ℓ is closer to a market j than any location of T_2, in which case the bigger the set of locations to which ℓ is being added, the smaller the value of $d_j^{(2)}(\mathbf{s})$ and hence of (3).

Proof of Theorem 2.1: We follow the same four-step outline we used for atomic selfish routing games in Lecture 12 (see Section 1). Let $\mathbf{s}$ denote an arbitrary PNE and $\mathbf{s}^*$ a surplus-maximizing outcome. In the first step, we invoke the PNE hypothesis once per player, with the outcome $\mathbf{s}^*$ providing hypothetical deviations. That is, since $\mathbf{s}$ is a PNE,

$$\pi_i(\mathbf{s}) \geq \pi_i(s_i^*, \mathbf{s}_{-i}) \tag{5}$$

[2]Games that possess these three properties are sometimes called *basic utility games* [3].

[3]Note the similarity to VCG payments (Lecture 7). This equality implies that every location game is a potential game (see Exercises). In our proof of Theorem 2.1, we only need the inequality $\pi_i(\mathbf{s}) \geq V(\mathbf{s})-V(\mathbf{s}_{-i})$. Games satisfying this inequality and properties (P1) and (P3) are called *valid utility games* [3].

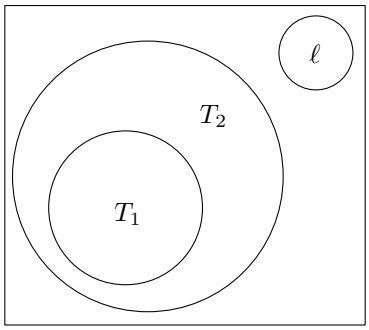


Figure 2: Submodularity: adding ℓ to T_2 yields a lower increase in value than adding ℓ to T_1 because $T_1 \subseteq T_2$.

for every player i. This is the only step of the proof that uses the assumption that $\mathbf{s}$ is a PNE.

The second step is to sum (5) over all the players, yielding

$$V(\mathbf{s}) \geq \sum_{i=1}^{k} \pi_i(\mathbf{s}) \geq \sum_{i=1}^{k} \pi_i({s_i}^*, \mathbf{s}_{-i}), \tag{6}$$

where the first inequality is property (P1) of location games.

The third step is to disentangle the final term of (6), and relate it to the only two quantities we really care about, $V(\mathbf{s})$ and $V(\mathbf{s}^*)$. By property (P2) of location games, we have

$$\sum_{i=1}^{k} \pi_i({s_i}^*, \mathbf{s}_{-i}) = \sum_{i=1}^{k} \left[V(s_i^*, \mathbf{s}_{-i}) - V(\mathbf{s}_{-i}) \right]. \tag{7}$$

To massage the right-hand side into a telescoping sum, we add extra locations to the terms. By submodularity of $V(\cdot)$ (property (P3)), we have

$$V({s_i}^*, \mathbf{s}_{-i}) - V(\mathbf{s}_{-i}) \geq V(s_1^*, \ldots, s_i^*, \mathbf{s}) - V(s_1^*, \ldots, s_{i-1}^*, \mathbf{s}).$$

Thus, the right-hand side of (7) can be bounded below by

$$\sum_{i=1}^{k} \left[V(s_1^*, \ldots, s_i^*, \mathbf{s}) - V(s_1^*, \ldots, s_{i-1}^*, \mathbf{s}) \right] = V(s_1^*, \ldots, s_k^*, s_1, \ldots, s_k) - V(\mathbf{s}) \geq V(\mathbf{s}^*) - V(\mathbf{s}),$$

where the inequality follows from the monotonicity of $V(\cdot)$ (property (P3)). This completes the third step of the proof.

The fourth and final step of the proof is to solve for the POA. We've proved that

$$V(\mathbf{s}) \geq V(\mathbf{s}^*) - V(\mathbf{s}),$$

so

$$\frac{V(\mathbf{s})}{V(\mathbf{s}^*)} \geq \frac{1}{2}$$

and the POA is at least $\frac{1}{2}$. This completes the proof of Theorem 2.1. ■

3 Smooth Games

There is a general recipe for deriving POA bounds, which includes our analyses of atomic selfish routing games and location games as special cases. There are also many other examples of this recipe that we won't have time to talk about. The goal of formalizing this recipe is not generalization for its own sake; as we'll see, POA bounds established via this recipe are automatically "robust" in several senses.

The following definition is meant to abstract the third "disentanglement" step in the POA upper bound proofs for atomic selfish routing games and location games.

Definition 3.1 (Smooth Games)

1. A cost-minimization game is (λ, μ)-*smooth* if

$$\sum_{i=1}^{k} C_i(s_i^*, \mathbf{s}_{-i}) \leq \lambda \cdot \text{cost}(\mathbf{s}^*) + \mu \cdot \text{cost}(\mathbf{s}) \tag{8}$$

   for all strategy profiles $\mathbf{s}, \mathbf{s}^*$.[4] Here $\text{cost}(\cdot)$ is an objective function that satisfies $\text{cost}(\mathbf{s}) \leq \sum_{i=1}^{k} C_i(\mathbf{s})$ for every strategy profile $\mathbf{s}$.[5]

2. A payoff-maximization game is (λ, μ)-*smooth* if

$$\sum_{i=1}^{k} \pi_i(s_i^*, \mathbf{s}_{-i}) \geq \lambda \cdot V(\mathbf{s}^*) - \mu \cdot V(\mathbf{s}) \tag{9}$$

   for all strategy profiles $\mathbf{s}, \mathbf{s}^*$. Here $V(\cdot)$ is an objective function that satisfies $V(\mathbf{s}) \geq \sum_{i=1}^{k} \pi_i(\mathbf{s})$ for every strategy profile $\mathbf{s}$.[6]

Justifying a new definition requires examples and consequences. We have already seen two examples of classes of smooth games; two consequences are described in the next section. In Lecture 12, we proved that every atomic selfish routing game with affine cost functions is a $(\frac{5}{3}, \frac{1}{3})$-smooth cost-minimization game. In Section 2, we proved that every location game is a $(1, 1)$-smooth payoff-maximization game. The conditions (8) and (9) were established in the third, "disentanglement" steps of these proofs. At the time, we had in mind the case where $\mathbf{s}$ and $\mathbf{s}^*$ are a PNE and an optimal outcome of the game, respectively, but the corresponding algebraic manipulations never used those facts and hence apply more generally to all pairs of strategy profiles.

[4] As long as (8) holds for some optimal solution $\mathbf{s}^*$ and all strategy profiles $\mathbf{s}$, all of the consequences in Section 4 continue to hold.

[5] In atomic selfish routing games, this inequality holds with equality.

[6] This is property (P1) of location games.

4 Robust POA Bounds in Smooth Games

In a (λ, μ)-smooth cost-minimization game with $\mu < 1$, every PNE $\mathbf{s}$ has cost at most $\frac{\lambda}{1-\mu}$ times that of an optimal outcome $\mathbf{s}^*$. To see this, use the assumption that the objective function satisfies $\text{cost}(\mathbf{s}) \leq \sum_{i=1}^{k} C_i(\mathbf{s})$, the PNE condition (once per player), and the smoothness assumption to derive

$$\text{cost}(\mathbf{s}) \leq \sum_{i=1}^{k} C_i(\mathbf{s}) \leq \sum_{i=1}^{k} C_i(s_i^*, \mathbf{s}_{-i}) \leq \lambda \cdot \text{cost}(\mathbf{s}^*) + \mu \cdot \text{cost}(\mathbf{s}), \tag{10}$$

and rearrange terms. Similarly, every PNE of a (λ, μ)-smooth payoff-maximization game has objective function value at least $\frac{\lambda}{1+\mu}$ times that of an optimal outcome. These are generalizations of our POA bounds of $\frac{5}{2}$ and $\frac{1}{2}$ for atomic selfish routing games with affine cost functions and location games, respectively.

We next describe two senses in which the POA bound of $\frac{\lambda}{1-\mu}$ or $\frac{\lambda}{1+\mu}$ for a (λ, μ)-smooth game is "robust." For the first, we recall last lecture's definition of coarse correlated equilibria (CCE).

Definition 4.1 A distribution σ on the set $S_1 \times \cdots \times S_k$ of outcomes of a cost-minimization game is a *coarse correlated equilibrium (CCE)* if for every player $i \in \{1, 2, \ldots, k\}$ and every unilateral deviation $s_i' \in S_i$,

$$\mathbf{E}_{\mathbf{s} \sim \sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s} \sim \sigma}[C_i(s_i', \mathbf{s}_{-i})]\,. \tag{11}$$

The equilibrium condition in (11) compares the expected cost of playing according to the distribution σ to that of an unconditional deviation. In Lecture 17, we'll show that simple and natural learning algorithms drive the time-averaged history of joint play to the set of CCE. In this sense, CCE are a quite tractable set of equilibria, and hence a relatively plausible prediction of realized play. See also Figure 3.

A drawback of enlarging the set of equilibria is that the POA, which is defined via a worst-case equilibrium, can only get worse. In smooth games, however, CCE are a "sweet spot" — permissive enough to be highly tractable, and stringent enough to enable good worst-case bounds.

Theorem 4.2 ([1]) *In every (λ, μ)-smooth cost-minimization game, the POA of CCE is at most $\frac{\lambda}{1-\mu}$.*

That is, the exact same POA bound that we derived in (10) for PNE holds more generally for all CCE. Similarly, in (λ, μ)-smooth payoff-maximization games, the POA bound of $\frac{\lambda}{1+\mu}$ applies more generally to all CCE (the details are left as an exercise). Our POA bounds of $\frac{5}{2}$ and $\frac{1}{2}$ for atomic selfish routing games and location games, respectively, may have seemed specific to PNE at the time, but since the proofs established the stronger smoothness condition (Definition 3.1), these POA bounds hold for all CCE.

Given the definitions, the proof of Theorem 4.2 is not difficult; let's just follow our nose.

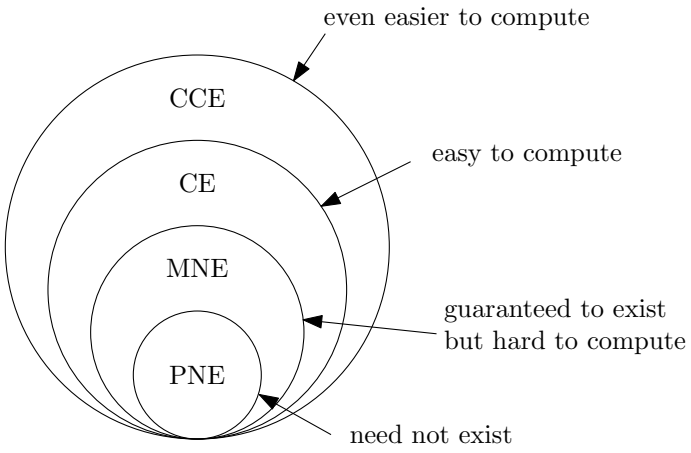


Figure 3: The Venn-diagram of the hierarchy of equilibrium concepts.

Proof of Theorem 4.2: Consider a (λ, μ)-smooth cost-minimization game, a coarse correlated equilibrium σ, and an optimal outcome $\mathbf{s}^*$. We can write

$$\begin{aligned}
\mathbf{E}_{\mathbf{s}\sim\sigma}[\text{cost}(\mathbf{s})] &\leq \mathbf{E}_{\mathbf{s}\sim\sigma}\left[\sum_{i=1}^{k} C_i(\mathbf{s})\right] && (12)\\
&= \sum_{i=1}^{k} \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] && (13)\\
&\leq \sum_{i=1}^{k} \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i^*, \mathbf{s}_{-i})] && (14)\\
&= \mathbf{E}_{\mathbf{s}\sim\sigma}\left[\sum_{i=1}^{k} C_i(s_i^*, \mathbf{s}_{-i})\right] && (15)\\
&\leq \mathbf{E}_{\mathbf{s}\sim\sigma}[\lambda \cdot \text{cost}(\mathbf{s}^*) + \mu \cdot \text{cost}(\mathbf{s})] && (16)\\
&= \lambda \cdot \text{cost}(\mathbf{s}^*) + \mu \cdot \mathbf{E}_{\mathbf{s}\sim\sigma}[\text{cost}(\mathbf{s})]\,, && (17)
\end{aligned}$$

where inequality (12) follows from the assumption on the objective function, equalities (13), (15), and (17) follow from linearity of expectation, inequality (14) follows from the definition (11) of a coarse correlated equilibrium (applied once per player i, with the hypothetical deviation s_i^*), and inequality (16) follows from the assumption that the game is (λ, μ)-smooth. Rearranging terms completes the proof. ■

Theorem 4.2 is already quite satisfying — we now have good POA bounds for an equilibrium concept that is guaranteed to exist and is easy to compute. It turns out that smooth games have a number of other nice properties, as well. We conclude this lecture by noting that the POA bound of $\frac{\lambda}{1-\mu}$ or $\frac{\lambda}{1+\mu}$ for a smooth game applies automatically to approximate equilibria, with the bound degrading gracefully as a function of the approximation parameter. For instance, define an *ϵ-pure Nash equilibrium (ϵ-PNE)* of a cost-minimization game

as a strategy profile $\mathbf{s}$ in which no player can decrease its cost by more than a $(1+\epsilon)$ factor via a unilateral deviation:

$$C_i(\mathbf{s}) \leq (1+\epsilon) \cdot C_i(s_i', \mathbf{s}_{-i}) \tag{18}$$

for every i and $s_i' \in S_i$. Then, the following guarantee holds.

Theorem 4.3 *For every (λ, μ)-smooth cost-minimization game G, every $\epsilon < \frac{1}{\mu} - 1$, every ϵ-PNE $\mathbf{s}$ of G, and every outcome $\mathbf{s}^*$ of G,*

$$C(\mathbf{s}) \leq \frac{(1+\epsilon)\lambda}{1-\mu(1+\epsilon)} \cdot C(\mathbf{s}^*).$$

The proof of Theorem 4.3 is not difficult and we leave it as an exercise. Similar results hold for smooth payoff-maximization games, and for approximate versions of other equilibrium concepts.

To illustrate Theorem 4.3, consider atomic selfish routing games with affine cost functions, which are $(\frac{5}{3}, \frac{1}{3})$-smooth. Theorem 4.3 implies that every ϵ-PNE of such a game with $\epsilon < 2$ has expected cost at most $\frac{5+5\epsilon}{2-\epsilon}$ times that of an optimal outcome.

References

[1] T. Roughgarden. Intrinsic robustness of the price of anarchy. In *41st ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2009.

[2] É. Tardos and T. Wexler. Network formation games and the potential function method. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 19, pages 487–516. Cambridge University Press, 2007.

[3] A. Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–425, 2002.

CS364A: Algorithmic Game Theory
Lecture #15: Best-Case and Strong Nash Equilibria*

Tim Roughgarden†

November 11, 2013

1 Network Cost-Sharing Games

1.1 Lecture Themes

The *externality* caused by a player in a game is the difference between its own objective function value and its contribution to the social objective function value. The models we've looked at thus far have *negative* externalities, meaning that players cause more harm to the system then they realize (or choose to care about). In a routing game, for example, a player does not take into account the additional cost its presence causes for the other players using the edges in its path.

There are also important applications that exhibit *positive* externalities. You usually join a campus organization or a social network to derive personal benefit from it, but your presence (hopefully) also enriches the experience of other people in the same group. As a player, you're generally bummed to see new players show up in a game with negative externalities, and excited for the windfalls of new players in a game with positive externalities. The first theme of this lecture is the study of positive externalities, in a concrete model of network formation.

In the model we study, there will generally be multiple pure Nash equilibria. We're used to that, from routing and location games, but here different equilibria can have wildly different costs. This motivates confining attention to a subset of "reasonable" Nash equilibria that hopefully possesses two properties: first, better worst-case inefficiency bounds should hold for the subset than for all equilibria; second, there should be a plausible narrative as to why equilibria in the subset are more worthy of study than the others. No fully satisfactory approach to this problem is known in the model we study, but we'll cover two partially successful approaches.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

1.2 The Model

A *network cost-sharing game* takes place in a graph $G = (V, E)$, which can be directed or undirected, and each edge $e \in E$ carries a fixed cost $\gamma_e \geq 0$. There are k players. Player i has a source vertex $s_i \in V$ and a sink vertex $t_i \in V$, and its strategy set is the s_i-t_i paths of the graph. Outcomes of the game correspond to path vectors $(P_1, \ldots, P_k)$, with the semantics that the subnetwork $(V, \cup_{i=1}^{k} P_i)$ gets formed.

We think of γ_e as the fixed cost of building the edge e — laying down high-speed Internet fiber to a neighborhood, for example — and this cost is independent of the number of players that use the edge. Players' costs are defined edge-by-edge, like in routing games. If $f_e \geq 1$ players use an edge e in their chosen paths, then they are jointly responsible for the edge's fixed cost γ_e. In this lecture, we assume that this cost is split equally amongst the players. That is, in the language of cost-minimization games (Lecture 13), the cost $C_i(\mathbf{P})$ of player i in the outcome $\mathbf{P}$ is

$$C_i(\mathbf{P}) = \sum_{e \in P_i} \frac{\gamma_e}{f_e}, \tag{1}$$

where $f_e = |\{j \,:\, e \in P_j\}|$ is the number of players that choose a path that includes e. The global objective is simply to minimize the total cost of the formed network:

$$\text{cost}(\mathbf{P}) = \sum_{e \in E \,:\, f_e \geq 1} \gamma_e. \tag{2}$$

Note that, analogous to routing games, the objective function (2) can equally well be written as the sum $\sum_{i=1}^{k} C_i(\mathbf{P})$ of the players' costs.

Remark 1.1 This is a very stylized game-theoretic model of how a network might form. Many such models have been studied, but it is quite difficult to capture observed properties of real-world networks with an analytically tractable model. See Jackson [5] for a textbook treatment of network formation games.

1.3 Example: VHS or Betamax

Let's build our intuition for network cost-sharing games through a couple of examples. The first example demonstrates how tragic miscoordination can occur in games with positive externalities.

Consider the simple network in Figure 1, with k players with a common source s and sink t. One can interpret an edge as the adoption of a particular technology. For example, back in the 1980s, there were two new technologies enabling home movie rentals. Betamax was lauded by technology geeks as the better one — corresponding to the lower-cost edge in Figure 1 — and VHS was the other one. VHS grabbed a larger market share early on. Since coordinating on a single technology proved the primary driver in consumers' decisions — have the better technology is little consolation for being unable to rent anything from your corner store — Betamax was eventually driven to extinction.

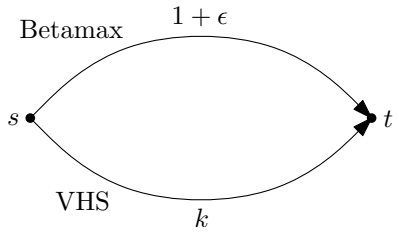


Figure 1: VHS or Betamax. The price of anarchy in a network cost-sharing game can be as large as the number k of players.

The optimal solution in Figure 1 is for all players to pick the upper edge, for a total cost of $1+\epsilon$. This is also a Nash equilibrium. Unfortunately, there is a second Nash equilibrium, in which all players pick the lower edge. Since the cost of k is split equally, each player pays 1. If a player deviated unilaterally to the upper edge, it would pay the full cost $1+\epsilon$ of that edge and thus suffer a higher cost. This example shows that the price of anarchy in network cost-sharing games can be as high as k, the number of players. (For a matching upper bound, see the Exercises.)

The VHS or Beta example is exasperating. We proposed a reasonable network model capturing positive externalities, and the price of anarchy — which helped us reason about several models already — is distracted by a manifestly unreasonable Nash equilibrium and yields no useful information. Can we salvage our approach by focusing only on the "reasonable" equilibria? We'll return to this question after considering another important example.

1.4 Example: Opting Out

Consider the network cost-sharing game shown in Figure 2. The k players have distinct sources $s_1, \ldots, s_k$ but a common destination t. They have the option of meeting at the rendezvous point v and continuing together to t, incurring a joint cost of $1+\epsilon$. Each player can also "opt out," meaning take the direct s_i-t path solo. (Insert your own joke about public transportation in California here.) Player i incurs a cost of $1/i$ for its opt-out strategy.

The optimal solution is clear: if all players travel through the rendezvous point, the overall cost is $1+\epsilon$. Unfortunately, this is not a Nash equilibrium: player k would pay slightly less by switching to its opt-out strategy (which is a dominant strategy for this player). Given that player k does not use the rendezvous in a Nash equilibrium, player $k-1$ does not either — it would have to pay at least $(1+\epsilon)/(k-1)$ with player k absent, and its opt-out strategy is cheaper. Iterating this argument, there is no Nash equilibrium in which any player travels through v. Meanwhile, the outcome in which all players opt out is a Nash equilibrium.[1] The cost of this (unique) Nash equilibrium is the kth Harmonic number $\sum_{i=1}^{k} \frac{1}{i}$, which lies between $\ln k$ and $\ln k + 1$.

The price of anarchy in the opt-out example is $\mathcal{H}_k$, which is much smaller than in the VHS

[1]We just performed a procedure called the iterated deletion of dominated strategies. When a unique outcome survives this procedure, it is the unique Nash equilibrium.

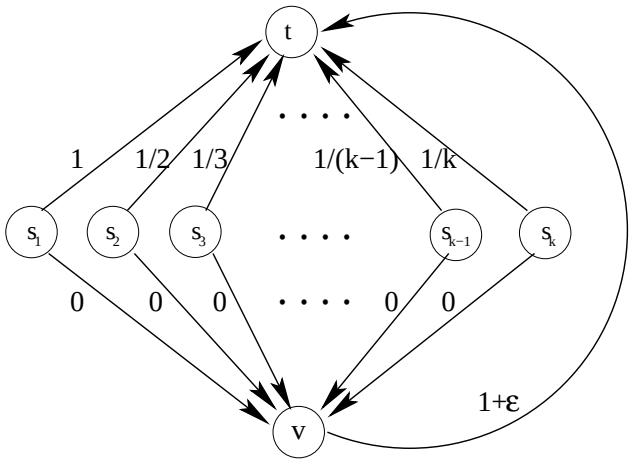


Figure 2: Opting out. There can be a unique Nash equilibrium with cost $\mathcal{H}_k$ times that of an optimal outcome.

or Betamax example, but it is also a much more compelling example. This inefficiency is not the result of multiple or unreasonable equilibria, and it captures the observable phenomenon that games with positive externalities can suffer efficiency loss from underparticipation.

2 The Price of Stability

The two examples in the previous section limit our ambitions: we cannot hope to prove anything interesting about worst-case Nash equilibria of network cost-sharing games, and even when there is a unique Nash equilibrium, it can cost $\mathcal{H}_k$ times that of an optimal solution. This section proves the following guarantee on *some* Nash equilibrium of every network cost-sharing game.

Theorem 2.1 (Price of Stability of Network Cost-Sharing Games [1]) *In every network cost-sharing game with k players, there exists a pure Nash equilibrium with cost at most $\mathcal{H}_k$ times that of an optimal outcome.*

The theorem asserts in particular that every network cost-sharing game possesses at least one pure Nash equilibrium, which is already a non-trivial fact. The opt-out example shows that the factor of $\mathcal{H}_k$ cannot be replaced by anything smaller.

The *price of stability* is the "optimistic" version of the price of anarchy, defined as the ratio between the cost of the best Nash equilibrium and that of an optimal outcome. Thus Theorem 2.1 asserts that the price of stability in every network cost-sharing game is at most $\mathcal{H}_k$. In terms of Figure 3, we are working with the entire set of pure Nash equilibria, but arguing only about one of them, rather than about all of them. This is the first occasion we've argued about anything other than the worst of a set of equilibria.

A bound on the price of stability, which only ensures that one equilibrium is approximately optimal, provides a significantly weaker guarantee than a bound on the price of

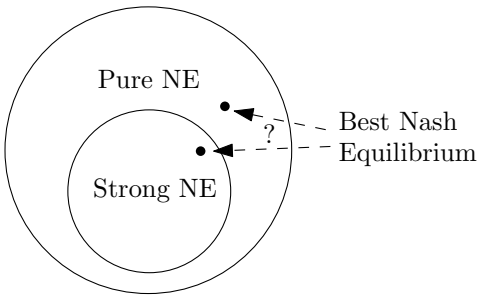


Figure 3: The best Nash equilibrium may be a strong NE, or not.

anarchy. It is well motivated in games where there is a third party that can propose an initial outcome — "default behavior" for the players. It's easy to find examples in real life where an institution or society effectively proposes one equilibrium out of many — even just in choosing which side of the road everybody drives on. For a computer science example, consider the problem of designing the default values of user-defined parameters of software or a network protocol. One sensible approach is to set default parameters so that users are not incentivized to change them and, subject to this, to optimize performance. The price of stability quantifies the necessary degradation in solution quality caused by restricting solutions to be equilibria.

Proof of Theorem 2.1: The proof of Theorem 2.1 goes through Rosenthal's potential function, introduced in Lecture 13. Recall the definition that we gave for atomic selfish routing games

$$\Phi(\mathbf{P}) = \sum_{e \in E} \sum_{i=1}^{f_e} c_e(i),$$

where c_e denotes the per-player cost incurred on edge e. Network cost-sharing games have exactly the same form as atomic selfish routing games — each of k players picks an s_i-t_i path in a network, and the player cost (1) is a sum of the edge costs, each of which depends only on the number of players using it — with the per-player cost of an edge e with f_e users being γ_e/f_e. Positive externalities are reflected by decreasing per-player cost functions, in contrast to the nondecreasing cost functions that were appropriate in routing games. Thus the potential function specializes to

$$\Phi(\mathbf{P}) = \sum_{e \in E} \sum_{i=1}^{f_e} \frac{\gamma_e}{i} = \sum_{e \in E} \gamma_e \sum_{i=1}^{f_e} \frac{1}{i} \tag{3}$$

in a network cost-sharing game.

In Lecture 13, we proved that the outcome that minimizes the potential function Φ is a Nash equilibrium, and we noted at the time that the proof worked for any cost functions, nondecreasing or not. That is, the strategic players are inadvertently striving to minimize Φ. This argument proves that every network cost-sharing game has a pure Nash equilibrium — the outcome that minimizes (3). For instance, in the VHS or Betamax example, the low-cost

Nash equilibrium minimizes (3) while the high-cost Nash equilibrium does not. While the minimizer of the potential function need not be the lowest-cost Nash equilibrium (see the Problems), we can prove that it has cost at most $\mathcal{H}_k$ times that of an optimal outcome.

The key observation is that the potential function in (3), whose numerical value we don't care about per se, approximates well the objective function (2) that we do care about. Precisely, since $\gamma_e \leq \gamma_e \sum_{i=1}^{f_e} \frac{1}{i} \leq \gamma_e \cdot \mathcal{H}_k$ for every $i \in \{1, 2, \ldots, k\}$, we have

$$\text{cost}(\mathbf{P}) \leq \Phi(\mathbf{P}) \leq \mathcal{H}_k \cdot \text{cost}(\mathbf{P}) \tag{4}$$

for every outcome $\mathbf{P}$. The inequalities (4) state that Nash equilibria are effectively trying to minimize an approximately correct function Φ, so it makes sense that one such equilibrium should approximately minimize the correct objective function.

To be precise, let $\mathbf{P}$ minimize Φ (a Nash equilibrium) and let $\mathbf{P}^*$ be an outcome of minimum cost. We have

$$\begin{aligned} \text{cost}(\mathbf{P}) &\leq \Phi(\mathbf{P}) \\ &\leq \Phi(\mathbf{P}^*) \\ &\leq \mathcal{H}_k \cdot \text{cost}(\mathbf{P}^*), \end{aligned}$$

where the first and last inequalities follow from (4) and the middle inequality follows from the choice of $\mathbf{P}$ as the minimizer of Φ. This completes the proof of Theorem 2.1. ■

Open Question 1 (POS in Undirected Networks) In the VHS or Betamax example, it doesn't matter whether the network is directed or undirected. The opt-out example, on the other hand, makes crucial use of a directed network (see the Exercises). An interesting open question is whether or not the price of stability of every undirected network cost-sharing game is bounded above by a constant; see [2] for the latest progress.

3 Strong Nash Equilibria and Their POA

This section gives an alternative approach to eluding the bad Nash equilibrium of the VHS or Betamax example and proving meaningful bounds on the inefficiency of equilibria in network cost-sharing games. The plan is to once again argue about all (i.e., worst-case) equilibria, but to first identify a strict subset of the pure Nash equilibria that we care about.

In general, when studying the inefficiency of equilibria in a class of games, one should zoom out (i.e., enlarge the set of equilibria) as much as possible subject to the existence of meaningful POA bounds. We zoomed out in games with negative externalities, such as routing and location games. The POA of pure Nash equilibria is close to 1 in these games, so we focused on extending worst-case bounds to ever-larger sets of equilibria. This lecture, where worst-case Nash equilibria can be highly suboptimal, we need to zoom in to recover interesting inefficiency bounds. In terms of Figure 3, we aim to bound the cost of all Nash equilibria that fall into the smaller set.

To motivate the subclass of Nash equilibria that we study, let's return to the VHS or Betamax example. The high-cost Nash equilibrium is an equilibrium because a player that deviates unilaterally would pay the full cost $1 + \epsilon$ of the upper edge. What if a coalition of *two* players deviated jointly to the upper edge? Each deviating player would be responsible for a cost of only $\approx \frac{1}{2}$, so this would be a profitable deviation. Thus the high-cost Nash equilibrium does not persist when coalitional deviations are allowed.

Definition 3.1 (Strong Nash Equilibrium) Let $\mathbf{s}$ be an outcome of a cost-minimization game.

(a) Strategies $\mathbf{s}'_A \in \prod_{i \in A} S_i$ are a *beneficial deviation* for a subset A of players if

$$C_i(\mathbf{s}'_A, \mathbf{s}_{-A}) \leq C_i(\mathbf{s})$$

for every player $i \in A$, with the inequality holding strictly for at least one player of A.

(b) The outcome $\mathbf{s}$ is a *strong Nash equilibrium* if there is no coalition of players with a beneficial deviation.

Nash equilibria can be thought of as strong Nash equilibria in which only singleton coalitions are allowed. Every strong Nash equilibrium is thus a Nash equilibrium — that is, the former concept is an *equilibrium refinement* of the latter.

To get a better feel for strong Nash equilibria, let's return to our two examples. As noted above, the high-cost Nash equilibrium of the VHS or Betamax example is not strong. The low-case Nash equilibrium is strong. In fact, since a coalition of the entire player set is allowed, intuition might suggest that strong Nash equilibria have to be optimal. This is the case when all players share the same source and destination (see the Exercises), but not in general. In the opt-out example, the same argument that proves that the outcome in which everybody "opt outs" is the unique Nash equilibrium also proves that it is a strong Nash equilibrium. Thus, the opt-out example has a strong Nash equilibrium with cost $\mathcal{H}_k$ times that of the minimum-cost outcome. Our next result states that no worse example is possible.

Theorem 3.2 (POA of Strong Nash Equilibria in Network Cost-Sharing Games [4]) *In every network cost-sharing game with k players, every strong Nash equilibrium has cost at most $\mathcal{H}_k$ times that of an optimal outcome.*

The guarantee in Theorem 3.2 differs from that in Theorem 2.1 in two ways. On the positive side, the guarantee holds for *every* strong Nash equilibrium, as opposed to just *one* Nash equilibrium. If it were the case that every network cost-sharing game has at least one strong Nash equilibrium, then Theorem 3.2 would be a strictly stronger statement than Theorem 2.1. The second difference, however, is that Theorem 3.2 does not assert existence, and for good reason (see Figure 4 below). These two differences render Theorems 2.1 and 3.2 incomparable guarantees.

Proof of Theorem 3.2: The proof bears some resemblance to our previous POA analyses, but has a couple of extra ideas. One nice feature is that — perhaps unsurprisingly given

the bound that we're trying to prove — the proof uses Rosenthal's potential function in an interesting way. Our previous POA analyses for classes of potential games (selfish routing, location games) did not make use of the potential function.

The first step in our previous POA analyses was to invoke the Nash equilibrium hypothesis once per player to generate upper bounds on players' equilibrium costs. Here, we're making a strong hypothesis — we begin by letting $\mathbf{P}$ be an arbitrary strong Nash equilibrium rather than an arbitrary Nash equilibrium — and aspire to a stronger conclusion. After all, what we're trying to prove is false for arbitrary Nash equilibria (as in the VHS or Betamax example).

The natural place to start is with the most powerful coalition $A_k = \{1, 2, \ldots, k\}$ of all k players. Why doesn't this coalition collectively switch to the optimal outcome $\mathbf{P}^*$? It must be that for some player i, $C_i(\mathbf{P}) \leq C_i(\mathbf{P}^*)$.[2] Rename the players so that this is player k.

We want an upper bound on the equilibrium cost of *every* player, not just that of player k. To ensure that we get an upper bound for a new player, we next invoke the strong Nash equilibrium hypothesis for the coalition $A_{k-1} = \{1, 2, \ldots, k-1\}$ — why don't these $k-1$ players collectively deviate to $\mathbf{P}^*_{A_{k-1}}$? There must be a player $i \in \{1, 2, \ldots, k-1\}$ with $C_i(\mathbf{P}) \leq C_i(\mathbf{P}^*_{A_{k-1}}, P_k)$. We rename the players of A_{k-1} so that this is true for player $k-1$ and continue.

Iterating the argument yields a renaming of the players as $\{1, 2, \ldots, k\}$ such that, for every i,

$$C_i(\mathbf{P}) \leq C_i(\mathbf{P}^*_{A_i}, \mathbf{P}_{-A_i}), \tag{5}$$

where $A_i = \{1, 2, \ldots, i\}$. Now that we have an upper bound on the equilibrium cost of every player, we can sum (5) over the players to obtain

$$\begin{aligned} \text{cost}(\mathbf{P}) &= \sum_{i=1}^{k} C_i(\mathbf{P}) \\ &\leq \sum_{i=1}^{k} C_i(\mathbf{P}^*_{A_i}, \mathbf{P}_{-A_i}) \qquad (6) \\ &\leq \sum_{i=1}^{k} C_i(\mathbf{P}^*_{A_i}). \qquad (7) \end{aligned}$$

Inequality (6) is immediate from (5). Inequality (7) follows from the fact that network cost-sharing games have positive externalities — deleting players only decreases the number f_e of players using a given edge and hence only increases the cost share of each remaining player on each edge. The motivation for the second inequality is to simplify our upper bound on the equilibrium cost to the point that it becomes a telescoping sum (cf., the location game analysis in Lecture 14).

Next we use the fact that network cost-sharing games are potential games. Recalling the definition (3) of the potential function Φ, we see that the decrease in potential function

[2] This inequality is strict if at least one other player is better off, but we won't need this stronger statement.

value from deleting a player is exactly the cost incurred by that player. Formally:

$$C_i(\mathbf{P}^*_{A_i}) = \sum_{e \in P^*_i} \frac{\gamma_e}{f^i_e} = \Phi(\mathbf{P}^*_{A_i}) - \Phi(\mathbf{P}^*_{A_{i-1}}), \tag{8}$$

where f^i_e denotes the number of players of A_i that use a path in $\mathbf{P}^*$ that includes edge e. This equation is the special case of the Rosenthal potential function condition (see Lecture 14) in which a player deviates to the empty-set strategy.

Combining (7) with (8), we obtain

$$\begin{aligned} \text{cost}(\mathbf{P}) &\leq \sum_{i=1}^{k} \Big[\Phi(\mathbf{P}^*_{A_i}) - \Phi(\mathbf{P}^*_{A_{i-1}})\Big] \\ &= \Phi(\mathbf{P}^*) \\ &\leq \mathcal{H}_k \cdot \text{cost}(\mathbf{P}^*), \end{aligned}$$

where the inequality follows from our earlier observation (4) that the potential function Φ can only overestimate the cost of an outcome by an $\mathcal{H}_k$ factor. This completes the proof of Theorem 3.2. ■

4 Epilogue

Network cost-sharing games can have "unreasonable" bad Nash equilibria, and this motivates the search for a subset of Nash equilibria with two properties: better worst-case bounds than for arbitrary Nash equilibria, and a plausible narrative justifying restricting the analysis to this subset. Both of our two solutions — best-case Nash equilibria and worst-case strong Nash equilibria — meet the first criterion, admitting an approximation bound of $\mathcal{H}_k$ rather than k. The justification for focusing on best-case Nash equilibria is strongest in settings where a third party can propose an equilibrium, although there is additional experimental evidence that potential function optimizers (as in Theorem 2.1) are more likely to be played than other Nash equilibria [3]. Worst-case bounds for strong Nash equilibria are attractive when such equilibria exist, as it is plausible that such equilibria are more likely to persist than regular Nash equilibria. While strong Nash equilibria are guaranteed to exist in classes of network cost-sharing games with sufficiently simple network structure [4], they do not, unfortunately, exist in general. Figure 4 gives a concrete example; we leave verifying the non-existence of strong Nash equilibria as an exercise.

References

[1] E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

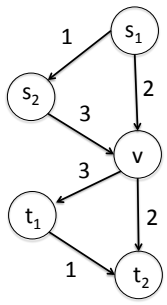


Figure 4: A network cost-sharing game with no strong Nash equilibrium.

[2] V. Bilò, M. Flammini, and L. Moscardelli. The price of stability for undirected broadcast network design with fair cost allocation is constant. In *Proceedings of the 54rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 638–647, 2013.

[3] R. Chen and Y. Chen. The potential of social identity for equilibrium selection. *American Economic Review*, 101(6):2562–2589, 2011.

[4] A. Epstein, M. Feldman, and Y. Mansour. Strong equilibrium in cost sharing connection games. *Games and Economic Behavior*, 67(1):51–68, 2009.

[5] M. O. Jackson. *Social and Economic Networks*. Princeton, 2008.

CS364A: Algorithmic Game Theory
Lecture #16: Best-Response Dynamics*

Tim Roughgarden†

November 13, 2013

1 Do Players Learn Equilibria?

In this lecture we segue into the third part of the course, which studies the following questions.

1. Do we expect strategic players do reach an equilibrium of a game, even in principle?
2. If so, will it happen quickly? As we'll see, theoretical computer science is well suited to contribute both positive and negative answers to this question.
3. If so, how does it happen? Which learning algorithms quickly converge to an equilibrium?

Affirmative answers to these questions are important because they justify equilibrium analysis. Properties of equilibria, such as a near-optimal objective function value, are not obviously relevant when players fail to find one. More generally, proving that natural learning algorithms converge quickly to an equilibrium lends plausibility to the predictive power of an equilibrium concept.

To reason about the three questions above, we require a behavioral model — "dynamics" — for players when not at an equilibrium. Thus far, we've just assumed that equilibria persist and that non-equilibria don't. This lecture focuses on variations of "best-response dynamics," while the next two lectures study dynamics based on regret-minimization.

No concrete model of learning will be fully convincing. We aspire toward simple and natural learning algorithms that lead to concrete and non-trivial predictions. Ideally, we would like to reach the same conclusion via multiple different learning processes. Then, even though we might not literally believe in any of the specific learning algorithms, we can have some confidence that the conclusion is robust, and not an artifact of the details of a particular learning process.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

2 Best-Response Dynamics

Best-response dynamics is a straightforward procedure by which players search for a pure Nash equilibrium (PNE) of a game. Specifically:

- While the current outcome $\mathbf{s}$ is not a PNE:
  - Pick an arbitrary player i and an arbitrary beneficial deviation s_i' for player i, and move to the outcome $(s_i', \mathbf{s}_{-i})$.

There might be many choices for the deviating player i and for the beneficial deviation s_i'. We leave both underspecified for the moment, specializing it later as needed.[1]

Best-response dynamics can only halt at a PNE — it cycles in any game without one. It can also cycle in games that have a PNE (see the Exercises).

Best-response dynamics is a perfect fit for *potential games*, discussed in passing in Lectures 13 and 15. Recall that a potential game admits a real-valued function Φ, defined on the outcomes of the game, such that for every outcome $\mathbf{s}$, every player i, and every unilateral deviation s_i' by i,

$$\Phi(s_i', \mathbf{s}_{-i}) - \Phi(\mathbf{s}) = C_i(s_i', \mathbf{s}_{-i}) - C_i(\mathbf{s}), \tag{1}$$

where C_i denotes player i's cost (or negative payoff) function. That is, a potential function tracks deviators' cost changes under unilateral deviations. Routing games (Lecture 12), location games (Lecture 14), and network cost-sharing games (Lecture 15) are all potential games.

We already know that potential games have PNE — the potential function minimizer is one. Best-response dynamics offer a more constructive proof of this fact.

Proposition 2.1 ([3]) *In a finite potential game, from an arbitrary initial outcome, best-response dynamics converges to a PNE.*

Proof: In every iteration of best-response dynamics, the deviator's cost strictly decreases. By (1), the potential function strictly decreases. Thus, no cycles are possible. Since the game is finite, best-response dynamics eventually halts, necessarily at a PNE. ■

Proposition 2.1 gives an affirmative answer to the first question of Section 1 for potential games — there is a natural procedure by which players can reach a PNE. Next we turn to the second question — how fast does this happen?

We consider three notions of "fast convergence," from strongest to weakest. The best-case scenario would be that best-response dynamics converges to a PNE in a polynomial number of iterations.[2] This strong conclusion is true when the potential function Φ can take on only

[1] This procedure is sometimes called "better-response dynamics," with the term "best-response dynamics" reserved for the version in which s_i' is chosen to maximize i's payoff (given $\mathbf{s}_{-i}$).

[2] Meaning polynomial in the number of players and the total number of players' strategies. The number of outcomes is exponential in the number of players — if there are n players each with two strategies, there are 2^n outcomes. Thus, the easy fact that the number of iterations of best-response dynamics is at most the number of outcomes of a potential game is not interesting in games with many players.

polynomially many distinct values (e.g., if it is integer-valued and polynomially bounded). In general, however, the potential function can take on exponentially many different values and best-response dynamics can decrease the potential function very slowly, requiring an exponential number of iterations to converge (see also Lecture 19).

3 Fast Convergence to ϵ-PNE in Symmetric Routing Games

The second notion of "fast convergence" settles for an approximate Nash equilibrium.

Definition 3.1 (ϵ-Pure Nash Equilibrium) For $\epsilon \in [0,1]$, an outcome $\mathbf{s}$ of a cost-minimization game is an *ϵ-pure Nash equilibrium (ϵ-PNE)* if, for every player i and deviation $s_i' \in S_i$,

$$C_i(s_i', \mathbf{s}_{-i}) \geq (1-\epsilon) \cdot C_i(\mathbf{s}). \tag{2}$$

This is essentially the same definition we used in Lecture 14, reparametrized for convenience. An ϵ-PNE in this lecture corresponds to a $\frac{1}{1-\epsilon}$-PNE in Lecture 14.

We next study *ϵ-best response dynamics*, in which we only permit moves that result in "significant" improvements. This is the key to converging much faster than under standard best-response dynamics. Precisely:

- While the current outcome $\mathbf{s}$ is not a ϵ-PNE:
  - Pick an arbitrary player i that has an *ϵ-move* — a deviation s_i' with $C_i(s_i', \mathbf{s}_{-i}) < (1-\epsilon)C_i(\mathbf{s})$ — and an arbitrary such move for the player, and move to the outcome $(s_i', \mathbf{s}_{-i})$.

ϵ-best response dynamics can only halt at an ϵ-PNE, and it eventually converges in every finite potential game. But how quickly?

Theorem 3.2 (Convergence of ϵ-Best Response Dynamics [2]) *Consider an atomic selfish routing game where:*

1. *All players have a common source vertex and a common sink vertex.*
2. *Cost functions satisfy the "α-bounded jump condition," meaning $c_e(x+1) \in [c_e(x), \alpha \cdot c_e(x)]$ for every edge e and positive integer x.*
3. *The MaxGain variant of ϵ-best-response dynamics is used: in every iteration, among players with an ϵ-move available, the player who can obtain the biggest absolute cost decrease moves to its minimum-cost deviation.*

Then, an ϵ-PNE is reached in $(\frac{k\alpha}{\epsilon} \log \frac{\Phi(\mathbf{s}^0)}{\Phi_{\min}})$ iterations.

Non-trivial constructions show that the first two hypotheses are necessary — if either is dropped, all variants of best-response dynamics take an exponential number of iterations in the worst case [5]; see also Lecture 19. In the third hypothesis, it is important that ϵ-best-response dynamics is used instead of standard best-response dynamics, but the result continues to hold for all natural variations of ϵ-best-response dynamics (see Exercises). Essentially, the only requirement is that every player is given the opportunity to move sufficiently often [2].

The plan for proving Theorem 3.2 is to strengthen quantitatively the proof of Proposition 2.1 — to prove that every iteration of ϵ-best-response dynamics decreases the potential function by *a lot*. We need two lemmas. The first one guarantees the existence of a player with high cost; if this player is chosen to move in an iteration, then the potential function decreases significantly. The issue is that some other player might be chosen to move by ϵ-best-response dynamics. The second lemma, which is the one that needs the hypotheses in Theorem 3.2, proves that the player chosen to move has cost within an α factor of that of any other player. This is good enough for fast convergence.

Lemma 3.3 *In every outcome* $\mathbf{s}$*, there is a player* i *with* $C_i(\mathbf{s}) \geq \Phi(\mathbf{s})/k$.

Proof: Recall from Lecture 12 that in a selfish routing game, the objective function is

$$\text{cost}(\mathbf{s}) = \sum_{e \in E} f_e \cdot c_e(f_e),$$

where f_e is the number of players that choose a strategy including the edge e, while the potential function is

$$\Phi(\mathbf{s}) = \sum_{e \in E} \sum_{i=1}^{k} c_e(i).$$

Since cost functions are nondecreasing, $\Phi(\mathbf{s}) \leq \text{cost}(\mathbf{s})$ for every outcome $\mathbf{s}$.

Since $\text{cost}(\mathbf{s}) = \sum_{i=1}^{k} C_i(\mathbf{s})$ in a selfish routing game, some player has cost at least as large as the average, and

$$\max_{i=1}^{k} C_i(\mathbf{s}) \geq \frac{\text{cost}(\mathbf{s})}{k} \geq \frac{\Phi(\mathbf{s})}{k},$$

as claimed. ■

The next lemma effectively relates the cost of the deviating player in ϵ-best-response dynamics to those of the other players.

Lemma 3.4 *Suppose player* i *is chosen in the outcome* $\mathbf{s}$ *by MaxGain* ϵ*-best-response dynamics, and takes the* ϵ*-move* s_i'*. Then*

$$C_i(\mathbf{s}) - C_i(s_i', \mathbf{s}_{-i}) \geq \frac{\epsilon}{\alpha} C_j(\mathbf{s}) \tag{3}$$

for every other player j*.*

Note that the definition of an ϵ-move is that $C_i(\mathbf{s}) - C_i(s_i', \mathbf{s}_{-i}) \geq \epsilon C_i(\mathbf{s})$. Thus (3) states that, up to a factor of α, the cost decrease enjoyed by i is at least as large as that of any other player taking an ϵ-move (whether that player has an ϵ-move available or not).

Proof of Lemma 3.4: Fix the player j. If j has an ϵ-move — which decreases player j's cost by at least $\epsilon C_j(\mathbf{s})$ — then (3) holds, even without the α, simply because i was chosen over j in MaxGain dynamics.

The trickier case is when the player j has no ϵ-move available. We use here that all players have the same strategy set: if s_i' is such a great deviation for player i, why isn't it for player j as well? That is, how can it be that

$$C_i(s_i', \mathbf{s}_{-i}) \leq (1-\epsilon) C_i(\mathbf{s}) \tag{4}$$

while

$$C_j(s_i', \mathbf{s}_{-j}) \geq (1-\epsilon) C_j(\mathbf{s})? \tag{5}$$

A key observation is that the outcomes $(s_i', \mathbf{s}_{-i})$ and $(s_i', \mathbf{s}_{-j})$ have at least $k-1$ strategies in common — s_i', played by i in the former outcome and by j in the latter, and the $k-2$ fixed strategies played by players other than i and j. Since the two outcomes differ in only one chosen strategy, the load on every edge differs by at most one in the two outcomes. By the α-bounded jump condition in Theorem 3.2, the cost of every edge differs by at most a factor of α in the two outcomes. In particular:

$$C_j(s_i', \mathbf{s}_{-j}) \leq \alpha \cdot C_i(s_i', \mathbf{s}_{-i}). \tag{6}$$

Note that both sides of inequality (6) reference the player that chooses strategy s_i' (j on the left-hand side, i on the right).

The inequalities (4)–(6) are compatible only if $C_j(\mathbf{s}) \leq \alpha \cdot C_i(\mathbf{s})$. Combining this with (4) yields $C_i(\mathbf{s}) - C_i(s_i') \geq \epsilon \cdot C_i(\mathbf{s}) \geq \frac{\epsilon}{\alpha} \cdot C_j(\mathbf{s})$, as required. ■

Lemma 3.3 guarantees that there is always a player that, if chosen to make an ϵ-move, rapidly decreases the potential function. Lemma 3.4 extends the conclusion to the player that is actually chosen to move. It is now a simple matter to upper bound the number of iterations required for convergence.

Proof of Theorem 3.2: In an iteration of ϵ-best-response dynamics where player i switches to the (ϵ-move) s_i',

$$\begin{aligned}
\Phi(\mathbf{s}) - \Phi(s_i', \mathbf{s}_{-i}) &= C_i(\mathbf{s}) - C_i(s_i', \mathbf{s}_{-i}) && (7)\\
&\geq \frac{\epsilon}{\alpha} \cdot \max_{j=1}^{k} C_j(\mathbf{s}) && (8)\\
&\geq \frac{\epsilon}{\alpha k} \cdot \Phi(\mathbf{s}), && (9)
\end{aligned}$$

where equation (7) follows from the definition of a potential function (Lecture 14) and inequalities (8) and (9) follow from Lemmas 3.4 and 3.3, respectively.

The upshot is that every iteration of ϵ-best-response dynamics decreases the potential function by at least a factor of $(1-\frac{\epsilon}{\alpha k})$. Thus, every $\frac{k\alpha}{\epsilon}$ iterations decrease the potential function by a constant factor.[3] Since the potential function begins at the value $\Phi(\mathbf{s}^0)$ and cannot drop lower than $\Phi_{\min}$, ϵ-best-response dynamics converges in $O(\frac{k\alpha}{\epsilon}\log\frac{\Phi(\mathbf{s}^0)}{\Phi_{\min}})$ iterations. ■

Theorem 3.2 is good justification for performing equilibrium analysis in atomic selfish routing games with a shared source and sink: many variations of the natural ϵ-best-response dynamics converge quickly to an approximate equilibrium. Unfortunately, Theorem 3.2 cannot be extended much further. In atomic routing games with multiple sources and sinks, for example, ϵ-best-response dynamics can require an exponential number of iterations to converge, no matter how the deviating player and deviation in each iteration are chosen [5].

4 Fast Convergence to Low-Cost Outcomes in Smooth Potential Games

This section explores our third and final notion of "fast convergence": quickly reaching outcomes with objective function value *as good as if* players had successfully converged to an (approximate) equilibrium. This is a weaker guarantee — it does not imply convergence to an approximate equilibrium — but is still quite compelling. In situations where the primary reason for equilibrium analysis is a performance (i.e., POA) bound, this weaker guarantee is a costless surrogate for equilibrium convergence.

Weakening our notion of fast convergence enables positive results with significantly wider reach. The next result applies to all potential games that are smooth (Lecture 14), including routing games (with arbitrary sources, sinks, and cost functions) and location games.

Theorem 4.1 ([1, 4]) *Consider a (λ,μ)-smooth cost-minimization game[4] with a positive potential function Φ that satisfies $\Phi(\mathbf{s}) \leq cost(\mathbf{s})$ for every outcome s. Let $\mathbf{s}^0,\ldots,\mathbf{s}^T$ be a sequence generated by MaxGain best-response dynamics, $\mathbf{s}^*$ a minimum-cost outcome, and $\eta > 0$ a parameter. Then all but*

$$O\left(\frac{k}{\eta(1-\mu)}\log\frac{\Phi(\mathbf{s}^0)}{\Phi_{\min}}\right) \tag{10}$$

outcomes $\mathbf{s}^t$ satisfy

$$cost(\mathbf{s}^t) \leq \left(\frac{\lambda}{1-\mu}+\eta\right)\cdot cost(\mathbf{s}^*),$$

where $\Phi_{\min}$ is the minimum potential function value of an outcome and k is the number of players.

[3] To see this, note that $(1-x)^{1/x} \leq (e^{-x})^{1/x} = \frac{1}{e}$ for $x \in (0,1)$.

[4] Recall from Lecture 14 that this means that $\sum_{i=1}^k C_i(s_i^*, \mathbf{s}_{-i}) \leq \lambda\cdot\text{cost}(\mathbf{s}^*) + \mu\cdot\text{cost}(\mathbf{s})$ for every pair $\mathbf{s},\mathbf{s}^*$ of outcomes.

The dynamics in Theorem 4.1 differ from those in Theorem 3.2 only in that the restriction to ϵ-moves is dropped. That is, in each iteration, the player i and deviation s_i' are chosen to maximize $C_i(\mathbf{s}) - C_i(s_i', \mathbf{s}_{-i})$.

Theorem 4.1 states that for all but a small number of outcomes in the sequence, the cost is essentially as low as if best-response dynamics had already converged to a PNE. These "bad outcomes" need not be successive — since an iteration of best-response dynamics can strictly increase the overall cost, a "good outcome" can be followed by a bad one.

Proof of Theorem 4.1: In the proof of Theorem 3.2, we showed that every iteration of the dynamics significantly decreased the potential function, thus limiting the number of iterations. Here, the plan is to show that whenever there is a *bad state* $\mathbf{s}$ — one that fails to obey the guarantee in (10) — the potential function decreases significantly. This will yield the desired bound on the number of bad states.

For an outcome $\mathbf{s}^t$, define $\delta_i(\mathbf{s}^t) = C_i(\mathbf{s}^t) - C_i(s_i^*, \mathbf{s}_{-i}^t)$ as the cost decrease i experiences by switching to s_i^*, and $\Delta(\mathbf{s}^t) = \sum_{i=1}^k \delta_i(\mathbf{s}^t)$. The value $\delta(\mathbf{s}^t)$ is nonpositive when $\mathbf{s}^t$ is a PNE, but in general it can be positive or negative. Using this notation and the smoothness assumption, we can derive

$$\text{cost}(\mathbf{s}^t) \leq \sum_{i=1}^k C_i(\mathbf{s}^t) = \sum_{i=1}^k \left[C_i(s_i^*, \mathbf{s}_{-i}^t) + \delta_i(\mathbf{s}^t)\right] \leq \lambda \cdot \text{cost}(\mathbf{s}^*) + \mu \cdot \text{cost}(\mathbf{s}^t) + \sum_{i=1}^k \delta_i(\mathbf{s}^t),$$

and hence

$$\text{cost}(\mathbf{s}^t) \leq \frac{\lambda}{1-\mu} \cdot \text{cost}(\mathbf{s}^*) + \frac{1}{1-\mu} \sum_{i=1}^k \delta_i(\mathbf{s}^t).$$

This inequality is stating that the cost of an outcome $\mathbf{s}^t$ is large only when the amount $\sum_i \delta_i(\mathbf{s}^t)$ players have to gain by unilateral deviations is large. It reduces the theorem to proving that only $O(\frac{k}{\eta(1-\mu)} \log \frac{\Phi(\mathbf{s}^0)}{\Phi_{\min}})$ states $\mathbf{s}^t$ are bad. Along the lines of Theorem 3.2, we next prove that bad states lead to a large decrease in the potential function.

In a bad state $\mathbf{s}^t$, since Φ underestimates cost,

$$\Delta(\mathbf{s}^t) \geq \eta(1-\mu)\text{cost}(\mathbf{s}^t) \geq \eta(1-\mu)\Phi(\mathbf{s}^t).$$

If a player i chooses a best response in the outcome $\mathbf{s}^t$, its cost decreases by at least $\delta_i(\mathbf{s}^t)$. Thus, in a bad state $\mathbf{s}^t$, the cost of the player chosen by maximum-gain best-response dynamics decreases by at least $\frac{\eta(1-\mu)}{k}\Phi(\mathbf{s}^t)$. Since Φ is a potential function, $\Phi(\mathbf{s}^{t+1}) \leq (1 - \frac{\eta(1-\mu)}{k})\Phi(\mathbf{s}^t)$ whenever $\mathbf{s}^t$ is a bad state. This, together with the fact that Φ can only decrease in each iteration, implies that after $\frac{k}{\eta(1-\mu)}$ bad states the potential function decreases by a constant factor. This implies the claimed bound of $O(\frac{k}{\eta(1-\mu)} \log \frac{\Phi(\mathbf{s}^0)}{\Phi_{min}})$ bad states in all. ■

References

[1] B. Awerbuch, Y. Azar, A. Epstein, V. S. Mirrkoni, and A. Skopalik. Fast convergence to nearly optimal solutions in potential games. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 264–273, 2008.

[2] S. Chien and A. Sinclair. Convergence to approximate Nash equilibria in congestion games. *Games and Economic Behavior*, 71(2):315–327, 2011.

[3] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[4] T. Roughgarden. Intrinsic robustness of the price of anarchy. In *41st ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2009.

[5] A. Skopalik and B. Vöcking. Inapproximability of pure Nash equilibria. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2008.

CS364A: Algorithmic Game Theory
Lecture #17: No-Regret Dynamics*

Tim Roughgarden†

November 18, 2013

This lecture continues to study the questions introduced last time. Do strategic players reach an equilibrium of a game? How quickly? By what learning processes? Positive results on these questions justify equilibrium analysis, including bounds on the price of anarchy.

Last lecture focused on best-response dynamics. These dynamics are most relevant for potential games, which cover many but not all interesting applications. This lecture, we study a second fundamental class of learning dynamics — *no-regret dynamics*. An attractive feature of these dynamics is their rapid convergence to an approximate equilibrium — a coarse correlated equilibrium (Lecture 13), not generally a Nash equilibrium — in arbitrary games.

1 External Regret

1.1 The Model

Most of the this lecture studies the *regret-minimization problem*, which concerns a single decision-maker playing a game against an adversary. Section 3 connects this single-player theory to multi-player games and their coarse correlated equilibria.

Consider a set A of $n \geq 2$ actions. The setup is as follows.

- At time $t = 1, 2, \ldots, T$:
  - A decision-maker picks a mixed strategy p^t — that is, a probability distribution — over its actions A.
  - An adversary picks a cost vector $c^t : A \to [0, 1]$.[1]

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]The important assumption is that costs are bounded. See the Exercises for extensions of today's results to negative costs (i.e., payoffs) and to costs in $[0, c_{\max}]$ instead of in $[0, 1]$.

– An action a^t is chosen according to the distribution p^t, and the decision-maker incurs cost $c^t(a^t)$. The decision-maker learns the entire cost vector c^t, not just the realized cost $c^t(a^t)$.[2]

For example, A could represent different investment strategies, or different driving routes between home and work. When we return to multi-player games (Section 3), the action set will be the strategy set of a single player, and the cost vector will be induced by the strategies chosen by all of the other players.

1.2 Lower Bounds

We seek a "good" algorithm for online decision-making problems of the above type. But the setup above seems a bit unfair, no? The adversary is allowed to choose a cost function *after* the decision-maker has committed to a mixed strategy. This asymmetry motivates asking what kind of guarantee we could possibly hope for in such a model. We next consider three examples that show limitations on what is possible.

Example 1.1 (Impossibility w.r.t. the Best Action Sequence) There is no hope of comparing the cost of an online decision-making algorithm to the cost of the best action sequence in hindsight — the latter quantity $\sum_{t=1}^{T} \min_{a \in A} c^t(a)$ is simply too strong a benchmark.

For instance, suppose $A = 2$ and fix an arbitrary online decision-making algorithm. Each day t, the adversary chooses the cost vector c^t as follows: if the algorithm plays the first strategy with probability at least $\frac{1}{2}$ then c^t is $(1 \;\; 0)$; otherwise the cost vector is $(0 \;\; 1)$. The adversary has forced the expected cost of the algorithm to be at least $\frac{T}{2}$ while ensuring that the cost of the best action sequence in hindsight is 0.

Example 1.1 motivates the following important definitions. Rather than comparing the expected cost of an algorithm to that of the best action *sequence* in hindsight, we compare it to the cost incurred by the best *fixed action* in hindsight. That is, our benchmark will be $\min_{a \in A} \sum_{t=1}^{T} c^t(a)$ rather than $\sum_{t=1}^{T} \min_{a \in A} c^t(a)$.

Definition 1.2 The *(time-averaged) regret* of the action sequence $a^1, \ldots . a^T$ with respect to the action a is

$$\frac{1}{T}\left[\sum_{t=1}^{T} c^t(a^t) - \sum_{i=1}^{T} c^t(a)\right]. \tag{1}$$

In this lecture, "regret" will always refer to Definition 1.2. Next lecture we discuss another notion of regret.

Definition 1.3 (No-Regret Algorithm) Let $\mathcal{A}$ be an online decision-making algorithm.

[2]The *bandit model*, where the decision-maker only learns the realized cost, has also been studied extensively (e.g. [2]). The guarantees presented in this lecture carry over, with somewhat worse bounds and more complex algorithms, to the bandit model as well.

(a) An *adversary for* $\mathcal{A}$ is a function that takes as input the day t, the mixed strategies $p^1, \ldots, p^t$ produced by $\mathcal{A}$ on the first t days, and the realized actions $a^1, \ldots, a^{t-1}$ of the first $t-1$ days, and produces as output a cost vector $c^t : [0,1] \to A$.

(b) An online decision-making algorithm has *no (external) regret* if for every adversary for it, the expected regret (1) with respect to every action $a \in A$ is $o(1)$ as $T \to \infty$.

Remark 1.4 (Combining Expert Advice) The problem of designing a no-regret algorithm is sometimes called "combining expert advice" — if we think of each action an "expert" that makes recommendations, then a no-regret algorithms performs asymptotically as well as the best expert.

Remark 1.5 (Adaptive vs. Oblivious Adversaries) The adversary in Definition 1.3 is sometimes called an *adaptive adversary*. An *oblivious adversary* is the special case in which the cost vector c^t depends only on t (and on the algorithm $\mathcal{A}$).

For this lecture, we'll adopt the no-regret guarantee of Definition 1.3 as the "holy grail" in the design of online decision-making algorithms. The first reason is that, as we'll see in Section 2, this goal can be achieved by simple and natural learning algorithms. The second reason is that the goal is non-trivial: as the examples below make clear, some ingenuity is required to achieve it. The third reason is that, when we pass to multi-player games in Section 3, no-regret guarantees will translate directly to coarse correlated equilibrium conditions.

Remark 1.6 In regret-minimization, one usually thinks of the number n of actions as fixed as the time horizon T tends to infinity. In a no-regret algorithm, the (time-averaged) regret can be a function of n but tends to 0 as the time horizon grows.

The next example rules out deterministic no-regret algorithms.

Example 1.7 (Randomization Is Necessary for No Regret) A simple consequence of the asymmetry between the decision-maker and the adversary is that there does not exist a no-regret deterministic algorithm. To see this, suppose there are $n \geq 2$ actions and fix a deterministic algorithm. At each time step t, the algorithm commits to a single action a^t. The obvious strategy for the adversary is to set the cost of action a^t to be 1, and the cost of every other action to be 0. Then, the cost of the algorithm is T while the cost of the best action in hindsight is at most $\frac{T}{n}$. Thus, even when there are only 2 actions, the algorithm has constant regret (as $T \to \infty$) with respect to some action a.

The next example does not rule out (randomized) no-regret algorithms, though it does limit the rate at which regret can vanish as the time horizon T grows.

Example 1.8 ($\Omega(\sqrt{(\ln n)/T})$ Regret Lower Bound) The next example shows that, even with only $n = 2$ actions, no (randomized) algorithm has expected regret vanishing faster than the rate $\Theta(1/\sqrt{T})$. A similar argument shows that, with n actions, expected regret cannot vanish faster than $\Theta(\sqrt{(\ln n)/T})$.

Consider an adversary that, independently on each day T, chooses uniformly at random between the cost vectors (1 0) and (0 1). No matter how smart or dumb an online decision-making algorithm is, its cumulative expected cost is exactly $\frac{T}{2}$. In hindsight, however, with constant probability one of the two fixed actions has cumulative cost only $\frac{T}{2} - \Theta(\sqrt{T})$. The reason is that if you flip T fair coins, while the expected number of heads is $\frac{T}{2}$, the standard deviation is $\Theta(\sqrt{T})$. This implies that there is a distribution over 2^T (oblivious) adversaries such that every algorithm has expected regret $\Omega(1/\sqrt{T})$, where the expectation is over the algorithm's coin flips and the choice of the adversary. It follows that for every algorithm, there exists an (oblivious) adversary for which the algorithm has expected regret $\Omega(1/\sqrt{T})$.

2 The Multiplicative Weights Algorithm

2.1 No-Regret Algorithms Exist

The most important result in this lecture is that *no-regret algorithms exist*. Next lecture we'll see that this fact alone has some amazing consequences. Even better, there are simple and natural such algorithms — while not a literal description of human player behavior, the guiding principles of the following algorithm are recognizable from the way many people learn and make decisions. Finally, the algorithm we discuss next has optimal regret, matching the lower bound provided by Example 1.8.

Theorem 2.1 ([3, 4], etc.) *There exist simple no-regret algorithms with expected regret $O(\sqrt{(\ln n)/T})$ with respect to every fixed action.*

An immediately corollary is that a logarithmic (in n) number of iterations suffice to drive the expected regret down to a small constant.

Corollary 2.2 *There exists an online decision-making algorithm that, for every $\epsilon > 0$, has expected regret at most ϵ with respect to every fixed action after $O((\ln n)/\epsilon^2)$ iterations.*

2.2 The Algorithm

We next discuss the *multiplicative weights (MW)* algorithm, which is also sometimes called *Randomized Weighted Majority* or *Hedge*. This algorithm has numerous applications beyond online decision-making [1]. Its design follows two guiding principles.

1. Past performance of actions should guide which action is chosen now. Since the action choice must be randomized (Example 1.7), the probability of choosing an action should be increasing in its past performance (i.e., decreasing in its cumulative cost).

2. Many instantiations of the above idea yield no-regret algorithms. For optimal regret bounds, however, it is important to aggressively punish bad actions — when a previously good action turns sour, the probability with which it is played should decrease at an exponential rate.

Here is a formal description of the MW algorithm. It maintains a weight, intuitively a "credibility," for each action. At each time step the algorithm chooses an action with probability proportional to its current weight. The weight of each action can only decrease, and the rate of decrease depends on the cost of the action.

1. Initialize $w^1(a) = 1$ for every $a \in A$.
2. For $t = 1, 2, \ldots, T$:
   (a) Play an action according to the distribution $p^t := w^t/\Gamma^t$, where $\Gamma^t = \sum_{a \in A} w^t(a)$ is the sum of the weights.
   (b) Given the cost vector c^t, decrease weights using the formula $w^{t+1}(a) = w^t(a) \cdot (1 - \epsilon)^{c^t(a)}$ for every action $a \in A$.[3]

For example, if all costs are either 0 or 1, then the weight of each action either stays the same (if $c^t(a) = 0$) or gets multiplied by $(1-\epsilon)$ (if $c^t(a) = 1$). We'll choose an exact value for ϵ later; it will be between 0 and $\frac{1}{2}$. For intuition, note that as ϵ grows small, the distribution p^t tends to the uniform distribution. Thus small values of ϵ encourage exploration. As ϵ tends to 1, the distribution p^t tends to the distribution that places all its mass on the action that has incurred the smallest cumulative cost so far. Thus large values of ϵ encourage "exploitation," and ϵ is a knob for interpolating between these two extremes. The MW algorithm is obviously simple to implement — the only requirement is to maintain a weight for each action.

2.3 The Analysis

This section proves the bound in Theorem 2.1 for the MW algorithm. We can restrict attention to oblivious adversaries (as defined in Remark 1.5) that fix a sequence of cost vectors $c^1, \ldots, c^T$ up front. The intuitive reason is that the behavior of the MW algorithm is independent of the realized actions $a^1, \ldots, a^t$: the distribution p^t chosen by the algorithm is a deterministic function of $c^1, \ldots, c^{t-1}$. Thus, there is no reason for a worst-case adversary for the MW algorithm to condition its cost vectors on previous realized actions. Similarly, a worst-case adversary does not need to explicitly condition on the distributions $p^1, \ldots, p^t$, since these are uniquely determined by the adversary's previous cost vectors $c^1, \ldots, c^{t-1}$.[4]

Fix an arbitrary sequence $c^1, \ldots, c^T$ of cost vectors. Recall $\Gamma^t = \sum_{a \in A} w^t(a)$ denotes the sum of the actions' weights at time t. The weight of every action (and hence Γ^t) can only decrease with t. The plan for the proof is to relate the two quantities we care about, the expected cost of the MW algorithm and the cost of the best fixed action, to Γ^T. The bound will then follow from some simple algebra and approximations.

[3] Other update steps, like $w^{t+1}(a) = w^t(a) \cdot (1 - \epsilon c^t(a))$, also work; see the Exercises.

[4] A bit more formally, one can solve for the worst adaptive adversary for the MW algorithm using backward induction, and the result is an oblivious adversary.

The first step is to show that if there is a good fixed action, then the weight of this action single-handedly shows that the final value Γ^T is pretty big. Formally, define $OPT := \sum_{t=1}^{T} c^t(a^*)$ as the cumulative cost for the best fixed action a^*. Then,

$$\begin{aligned}\Gamma^T &\geq w^T(a^*) \\ &= \underbrace{w^1(a^*)}_{=1} \prod_{t=1}^{T} (1-\epsilon)^{c^t(a^*)} \\ &= (1-\epsilon)^{OPT}.\end{aligned}$$

This connects our intermediate quantity Γ^T with one of the two quantities that we care about, namely OPT.

The second step, and the step which is special to the MW algorithm and its close cousins, is that the sum of weights Γ^t decreases exponentially fast with the expected cost of the MW algorithm. This implies that the algorithm can only incur large cost if all fixed actions are bad.

Precisely, the expected cost of the MW algorithm at time t is

$$\sum_{a \in A} p^t(a) \cdot c^t(a) = \sum_{a \in A} \frac{w^t(a)}{\Gamma^t} \cdot c^t(a). \tag{2}$$

Next, to understand Γ^{t+1} as a function of Γ^t and the expected cost (2), we write

$$\begin{aligned}\Gamma^{t+1} &= \sum_{a \in A} w^{t+1}(a) \\ &= \sum_{a \in A} w^t(a) \cdot (1-\epsilon)^{c^t(a)} \\ &\leq \sum_{a \in A} w^t(a) \cdot (1 - \epsilon c^t(a)) \qquad (3) \\ &= \Gamma^t(1 - \epsilon\nu^t),\end{aligned}$$

where (3) follows from the fact that $(1-\epsilon)^x \leq 1 - \epsilon x$ for $\epsilon \in [0, \frac{1}{2}]$ and $x \in [0,1]$ (see the Exercises), and ν^t denotes the expected cost (2) of the MW algorithm at time t. As promised, the expected algorithm cost (and ϵ) govern the rate at which the total weight Γ^t decreases.

Our goal is to upper bound the cumulative expected cost $\sum_{i=1}^{t} \nu^t$ of the MW algorithm in terms of OPT. We've related these quantities through Γ^T:

$$(1-\epsilon)^{OPT} \leq \Gamma^T \leq \underbrace{\Gamma^1}_{=n} \prod_{t=1}^{T} (1 - \epsilon\nu^t)$$

and hence

$$OPT \cdot \ln(1-\epsilon) \leq \ln n + \sum_{t=1}^{T} \ln(1 - \epsilon\nu^t).$$

We want to extract the ν^t's from the clutches of the logarithm, so it makes sense to recall the Taylor expansion

$$\ln(1-x) = -x - \tfrac{x^2}{2} - \tfrac{x^3}{3} - \cdots .$$

By throwing away all (negative) terms but the first, we can use this expansion to upper bound $\ln(1-\epsilon\nu^t)$ by $-\epsilon\nu^t$. While we're at it, we may as well lower bound $\ln(1-\epsilon)$ by throwing out all terms but the first two, and doubling the second term to compensate (assuming here that $\epsilon \leq \frac{1}{2}$), yielding $-\epsilon - \epsilon^2$. Summarizing, for $\epsilon \in (0, \frac{1}{2}]$ we have

$$OPT \cdot \left[-\epsilon - \epsilon^2\right] \leq \ln n + \sum_{t=1}^{T} (-\epsilon\nu^t)$$

and hence

$$\sum_{t=1}^{T} \nu^t \leq OPT \cdot (1+\epsilon) + \frac{\ln n}{\epsilon} \leq OPT + \epsilon T + \frac{\ln n}{\epsilon}, \tag{4}$$

where in the second inequality we use the very weak upper bound that, since costs are at most 1, OPT is at most T.

We now set the free parameter ϵ in the MW algorithm to equalize the two error terms in (4) — that is, to $\sqrt{\ln n/T}$. Then, the cumulative expected cost of the MW algorithm is at most $2\sqrt{T \ln n}$ more than the cumulative cost of the best fixed action; dividing both sides by T shows that (per-time-step) regret is at most $2\sqrt{\ln n/T}$. This completes the proof of Theorem 2.1.

Remark 2.3 (When T Is Unknown) In setting the parameter ϵ, we assumed that the time horizon T is known a priori. When this is not the case, the algorithm can be modified as follows: at day t, use the value $\epsilon = \sqrt{\ln n/\hat{T}}$, where $\hat{T}$ is the smallest power of 2 larger than t. The regret guarantee of Theorem 2.1 carries over to this algorithm (see the Exercises).

Recall that Example 1.8 shows that Theorem 2.1 is optimal up the constant in the additive term. Corollary 2.2 is also worth remembering — only $\frac{4 \ln n}{\epsilon^2}$ iterations of the MW algorithm are necessary to achieve expected regret at most ϵ.

3 No-Regret Dynamics

We now pass from single-player to multi-player settings. We use the language of cost-minimization games (Lecture 13); there is an obvious analog for payoff-maximization games. In each time step $t = 1, 2, \ldots, T$ of *no-regret dynamics*:

1. Each player i simultaneously and independently chooses a mixed strategy p_i^t using a no-regret algorithm.

2. Each player i receives a cost vector c_i^t, where $c_i^t(s_i)$ is the expected cost of strategy s_i when the other players play their chosen mixed strategies. That is, $c_i^t(s_i) = \mathbf{E}_{\mathbf{s}_{-i} \sim \sigma_{-i}}[C_i(s_i, \mathbf{s}_{-i})]$, where $\sigma_{-i} = \prod_{j \neq i} \sigma_j$.

No-regret dynamics are well defined because no-regret algorithms exist (Theorem 2.1). Each player can use whatever no-regret algorithm it wants. Players move simultaneously, unlike in best-response dynamics, but the following results also extend to players that move sequentially (see the Exercises).

No-regret dynamics can be implemented very efficiently. If each player uses the MW algorithm, for example, then in each iteration each player does a simple update of one weight per strategy, and only $O(\frac{\ln n}{\epsilon^2})$ iterations of this are required before every player has expected regret at most ϵ for every strategy. (Here n is the maximum size of a player's strategy set.) The next result is simple but important: the time-averaged history of joint play under no-regret dynamics converges to the set of coarse correlated equilibrium — the biggest set in our hierarchy of equilibria (Lecture 13). This forges a fundamental connection between a static equilibrium concept and outcomes generated by natural learning dynamics.

Proposition 3.1 *Suppose after T iterations of no-regret dynamics, every player of a cost-minimization game has regret at most ϵ for each of its strategies. Let $\sigma^t = \prod_{i=1}^k p_i^t$ denote the outcome distribution at time t and $\sigma = \frac{1}{T}\sum_{t=1}^T \sigma^t$ the time-averaged history of these distributions. Then σ is an ϵ-approximate coarse correlated equilibrium, in the sense that*

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i', \mathbf{s}_{-i})] + \epsilon$$

for every player i and unilateral deviation s_i'.

Proof: By definition, for every player i,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] = \frac{1}{T}\sum_{t=1}^T \mathbf{E}_{\mathbf{s}\sim\sigma^t}[C_i(\mathbf{s})] \tag{5}$$

and

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(s_i', \mathbf{s}_{-i})] = \frac{1}{T}\sum_{t=1}^T \mathbf{E}_{\mathbf{s}\sim\sigma^t}[C_i(s_i', \mathbf{s}_{-i})]\,. \tag{6}$$

The right-hand sides of (5) and (6) are the time-averaged expected costs of player i when playing according to its no-regret algorithm and when playing the fixed action s_i' every day, respectively. Since every player has regret at most ϵ, the former is at most ϵ more than the latter. This verifies the approximate coarse correlated equilibrium condition. ■

Remark 3.2 For Proposition 3.1, it is important that the decision-making algorithms used by the players have no regret with respect to adaptive adversaries (Remark 1.5). The mixed strategy chosen by player i at time t affects the cost vectors c_j^t received by the other players $j \neq i$ at time t, hence it affects their chosen strategies at future time steps, and hence it affects the cost vectors received by player i at future time steps. That is, when other players are themselves using adaptive learning algorithms to choose strategies, they correspond to an adaptive adversary.

In Lecture 14 we proved that POA bounds for smooth games — including all of the examples we've studied in this course — automatically extend to coarse correlated equilibria. With an approximate equilibrium, the POA bounds remain approximately correct.

Corollary 3.3 *Suppose after T iterations of no-regret dynamics, player i has expected regret at most R_i for each of its actions. Then the time-averaged expected objective function value $\frac{1}{T}\mathbf{E}_{\mathbf{s}\sim\sigma^i}[cost(\mathbf{s})]$ is at most*

$$\frac{\lambda}{1-\mu}cost(\mathbf{s}^*) + \frac{\sum_{i=1}^k R_i}{1-\mu}.$$

In particular, as $T \to \infty$, $\sum_{i=1}^k R_i \to 0$ and the guarantee converges to the standard POA bound $\frac{\lambda}{1-\mu}$. We leave the proof of Corollary 3.3 as an exercise.

4 Epilogue

The key take-home points of this lecture are:

1. Very simple learning algorithms lead remarkably quickly to (approximate) coarse correlated equilibria (CCE).

2. In this sense, CCE are unusually tractable, and hence unusually plausible as a prediction for player behavior.

3. Since POA bounds in smooth games apply to no-regret dynamics, they are particularly robust to the behavioral model.

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[3] J. Hannan. Approximation to Bayes risk in repeated play. *Ccontributions to the Theory of Games*, 3:97–139, 1957.

[4] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

CS364A: Algorithmic Game Theory
Lecture #18: From External Regret to Swap Regret and the Minimax Theorem*

Tim Roughgarden†

November 20, 2013

1 Swap Regret and Correlated Equilibria

Last lecture we proved that coarse correlated equilibria (CCE) are tractable, in a satisfying sense: there are simple and computationally efficient learning procedures that converge quickly to the set of CCE. Of course, if anything in our equilibrium hierarchy (Figure 1) was going to be tractable, it was going to be CCE, the biggest set.

The good researcher is never satisfied and always seeks stronger results. What can we say if we zoom in to the next-biggest set, the correlated equilibria? The first part of this lecture shows that correlated equilibria are also tractable. We'll give computationally efficient — if not quite as simple — learning procedures that converge fairly quickly to this set.

Remark 1.1 (Learning vs. Linear Programming) The computational tractability of correlated and coarse correlated equilibria — and mixed Nash equilibria of two-player zero-sum games, see Section 3 — can also be demonstrated by formulating linear programs for them. A bonus of the linear programming approach is that an exact, rather than an approximate, equilibrium can be computed in polynomial time. Another advantage is that linear optimization over the set of equilibria remains computationally tractable, while learning procedures merely guide behavior to somewhere in the set. On the other hand, exact linear programming algorithms seem wholly unrelated to any reasonable model of how agents learn in games.

Recall from Lecture 13 and Exercise 59 that a *correlated equilibrium* of a cost-minimization game is a distribution σ over outcomes such that, for every player i with strategy set S_i and every switching function $\delta : S_i \to S_i$,

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\delta(s_i), \mathbf{s}_{-i})]\,.$$

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

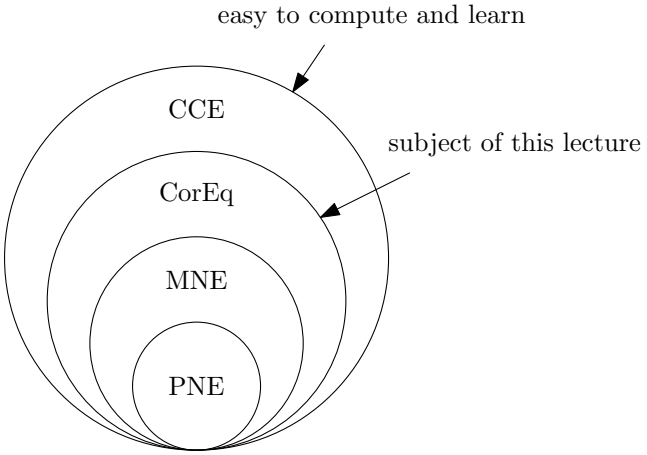


Figure 1: The hierarchy of equilibria from Lecture 13.

For example, in the "traffic intersection game" of Lecture 13, mixing 50/50 between the two pure Nash equilibria gives a (non-Nash) correlated equilibria.

Recall the online decision-making setting from last time: every day $t = 1, 2, \ldots, T$, a decision-maker commits to a distribution p^t over its n actions A, then an adversary chooses a cost function $c^t : A \to [0, 1]$, and finally an action a^t is chosen according to p^t, resulting in cost $c^t(a^t)$ to the decision-maker. Last lecture described an algorithm with time-averaged expected cost as small as that of every fixed action, up to an error term that goes to 0 as the time horizon T grows. When every player of a game uses such a no-external-regret algorithm to choose a strategy at each time step, the time-averaged history of joint play is an approximate CCE. Is there a more stringent regret notion that enjoys an analogous correspondence with correlated equilibria?

Definition 1.2 An online decision-making algorithm has *no swap regret* if for every adversary for it, the expected swap regret

$$\frac{1}{T}\left[\sum_{t=1}^{T} c^t(a^t) - \sum_{i=1}^{T} c^t(\delta(a^t))\right] \tag{1}$$

with respect to every switching function $\delta : A \to A$ is $o(1)$ as $T \to \infty$.

Because fixed actions are the special case of constant switching functions, an algorithm with no swap regret also has no external regret.

In each time step t of *no-swap-regret dynamics*, every player i independently chooses a mixed strategy p_i^t according to a no-swap-regret algorithm. Cost vectors are defined as in no-regret dynamics: $c_i^t(s_i)$ is the expected cost of strategy $s_i \in S_i$, given that every other player j plays its chosen mixed strategy p_j^t. The connection between correlated equilibria and no-swap-regret dynamics is the same as that between CCE and no-(external-)regret dynamics.

Proposition 1.3 *Suppose after T iterations of no-swap-regret dynamics, every player of a cost-minimization game has swap regret at most ϵ for each of its switching functions. Let $\sigma^t = \prod_{i=1}^k p_i^t$ denote the outcome distribution at time t and $\sigma = \frac{1}{T}\sum_{t=1}^T \sigma^t$ the time-averaged history of these distributions. Then σ is an ϵ-approximate correlated equilibrium, in the sense that*

$$\mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\mathbf{s})] \leq \mathbf{E}_{\mathbf{s}\sim\sigma}[C_i(\delta(s_i), \mathbf{s}_{-i})] + \epsilon$$

for every player i and switching function $\delta : S_i \to S_i$.

2 A Black-Box Reduction From Swap Regret to External Regret

This section gives a "black-box reduction" from the problem of designing a no-swap-regret algorithm to that of designing a no-external-regret algorithm — a problem that we already solved in the previous lecture.

Theorem 2.1 ([1]) *If there is a no-external-regret algorithm, then there is a no-swap-regret algorithm.*

As we'll see, the reduction in Theorem 2.1 also preserves computational efficiency. For example, plugging the multiplicative weights algorithm into this reduction yields a polynomial-time no-swap-regret algorithm. We conclude that correlated equilibria are tractable in the same strong sense as coarse correlated equilibria.

Proof of Theorem 2.1: The reduction is very natural, one that you'd hope would work. It requires one clever trick, as we'll see at the end of the proof.

Let n denote the number of actions. Let $M_1, \ldots, M_n$ denote n different no-(external-)regret algorithms, for example n instantiations of the multiplicative weights algorithm. Each of these algorithms is poised to produce probability distributions over actions and receive cost vectors as feedback. Very roughly, we can think of algorithm M_j as responsible for protecting against profitable deviations from action j to other actions.

The "master algorithm" M is as follows; see also Figure 2.

1. At time $t = 1, 2, \ldots, T$:
   (a) Receive distributions $q_1^t, \ldots, q_n^t$ over actions from the algorithms $M_1, \ldots, M_n$.
   (b) Compute and output a consensus distribution p^t.
   (c) Receive a cost vector c^t from the adversary.
   (d) Give algorithm M_j the cost vector $p^t(j) \cdot c^t$.

We discuss how to compute the consensus distribution p^t from the distributions $q_1^t, \ldots, q_n^t$ at the end of the proof; this is the clever trick in the reduction. The fourth step parcels out

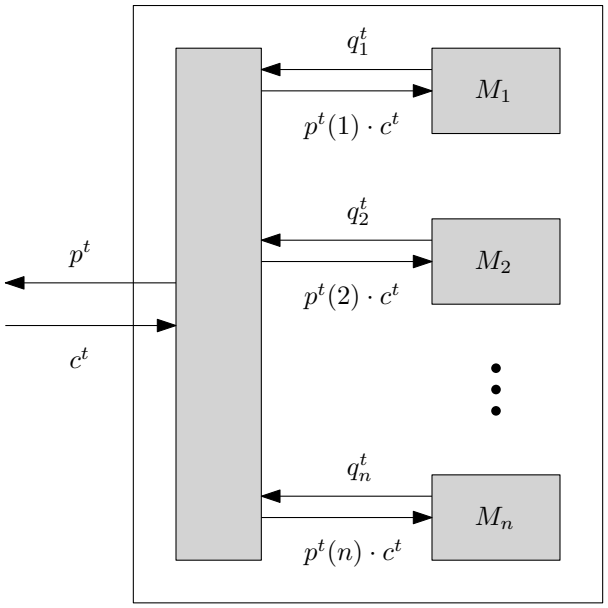


Figure 2: Blackbox reduction from swap regret to external regret.

the true cost vector c^t to the no-external-regret algorithms, scaled according to the current relevance (i.e., $p^t(j)$) of the algorithm.

Our hope is that we can piggyback on the no-external-regret guarantee provided by each algorithm M_j and conclude a no-swap-regret guarantee for the master algorithm M. Let's take stock of what we've got and what we want, parameterized by our computed consensus distributions $p^1, \ldots, p^T$.

The time-averaged expected cost of the master algorithm is

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} p^t(i) \cdot c^t(i). \tag{2}$$

The time-averaged expected cost under a switching function $\delta : A \to A$ is

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} p^t(i) \cdot c^t(\delta(i)). \tag{3}$$

Remember that our goal is to prove that (2) is at most (3), plus a term that goes to 0 as $T \to \infty$, for every switching function δ.

Now adopt the perspective of an algorithm M_j. This algorithm believes that actions are being chosen according to its recommended distributions $q_j^1, \ldots, q_j^T$ and that the true cost vectors are $p^1(j) \cdot c^1, \ldots, p^T(j) \cdot c^T$. Thus, the algorithm perceives its time-averaged expected cost as

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} q_j^t(i) \left(p^t(j) c^t(i) \right). \tag{4}$$

Since M_j is a no-regret algorithm, its perceived cost (4) is, up to the regret term, at most that of every fixed action $k \in A$:

$$\frac{1}{T}\sum_{t=1}^{T} p^t(j)c^t(k) + R_j, \tag{5}$$

where $R_j \to 0$ as $T \to \infty$.

Now fix a switching function δ. Summing the inequality between (4) and (5) over all $j = 1, 2, \ldots, n$, with k instantiated as $\delta(j)$ in (5), yields

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} q_j^t(i)p^t(j)c^t(i) \leq \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{n} p^t(j)c^t(\delta(j)) + \sum_{j=1}^{n} R_j. \tag{6}$$

Observe that the right-hand side of (6) is exactly (3), up to a term $\sum_{j=1}^{n} R_j$ that goes to 0 as $T \to \infty$. (Recall that we think of n as fixed as $T \to \infty$.) Indeed, we chose the splitting of the cost vector c^t amongst the no-external-regret algorithms $M_1, \ldots, M_n$ to guarantee this property.

If we can choose the consensus distributions $p^1, \ldots, p^T$ so that (2) and the left-hand side of (6) coincide, then the reduction will be complete. We show how to choose each p^t so that, for each $i \in A$ and $t = 1, 2, \ldots, T$,

$$p^t(i) = \sum_{j=1}^{n} q_j^t(i)p^t(j). \tag{7}$$

The left- and right-hand sides of (7) are the coefficients of $c^t(i)$ in (2) and in the left-hand side of (6), respectively.

The equations (7) might be familiar as those defining the stationary distribution of a Markov chain. This is the key trick in the reduction: given distributions $q_1^t, \ldots, q_n^t$ from algorithms $M_1, \ldots, M_n$ at time t, form the following Markov chain (Figure 3): the set of states is $A = \{1, 2, \ldots, n\}$, and for every $i, j \in A$, the transition probability from j to i is $q_j^t(i)$. That is, the distribution q_j^t specifies the transition probabilities out of state j. A probability distribution p^t satisfies (7) if and only if it is the stationary distribution of this Markov chain. At least one such distribution exists, and one can be computed in polynomial time via an eigenvector computation (see e.g. [3]). This completes the reduction. ■

Our choice of the consensus distribution p^t from the no-external-regret algorithms' suggestions $q_1^t, \ldots, q_n^t$ is uniquely defined by the proof approach, but it also has a natural interpretation as a limit of the following decision-making process. Suppose you first ask an arbitrary algorithm M_{j_1} for a recommended strategy. It gives you a recommendation j_2 drawn from its distribution $q_{j_1}^t$. You then ask algorithm M_{j_2} for a recommendation, which it draws from its distribution $q_{j_2}^t$, and so on. This random process is effectively trying to converge to a stationary distribution p^t of the Markov chain defined above, and will successfully do so when the chain is ergodic.

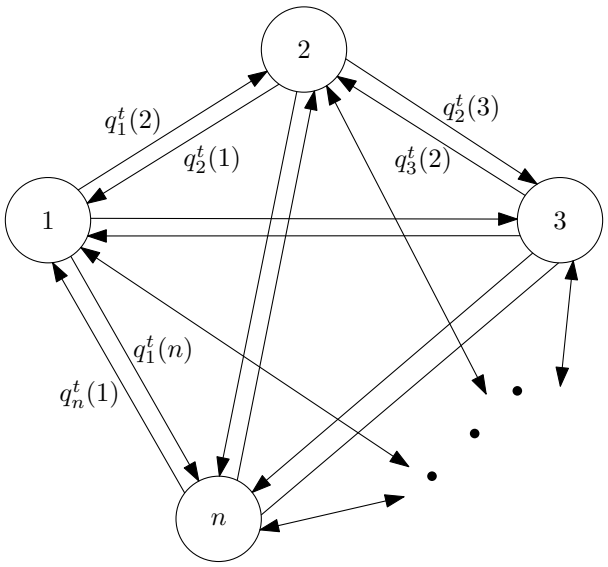


Figure 3: Markov chain.

3 The Minimax Theorem for Two-Player, Zero-Sum Games

Having resolved the complexity of correlated equilibria in satisfactory fashion, we now zoom in further to the set of mixed Nash equilibria (Figure 1). We'll see next week that, while the set of mixed Nash equilibria is guaranteed to be non-empty, computing one is a computationally intractable problem. Today we'll focus on a special case with a happier answer: two-player zero-sum games.

In a two-player zero-sum game, the payoff of each player is the negative of the other — one player can win only at the other's expense. Such a game can be specified by a single matrix A, with the two strategy sets corresponding to the rows and columns. The entry a_{ij} specifies the payoff of the row player in the outcome (i, j) and the negative payoff of the column player in this outcome. Thus, the row and column players prefer bigger and smaller numbers, respectively. The matrix below describes the payoffs in the Rock-Paper-Scissors game (Lecture 1) in our current language.

	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

Pure Nash equilibria generally don't exist in two-player zero-sum games, so the focus is squarely on mixed Nash equilibria. We use $\mathbf{x}$ and $\mathbf{y}$ to denote mixed strategies (probability distributions) over the rows and columns, respectively.

With mixed strategies, we think of each player as randomizing independently. Thus, the expected payoff of the row player when payoffs are given by A, the row strategy is $\mathbf{x}$, and the column strategy is $\mathbf{y}$, is

$$\sum_{i,j} \mathbf{Pr}_{\mathbf{x}}[i] \cdot \mathbf{Pr}_{\mathbf{y}}[j] \cdot a_{ij} = \mathbf{x}^T A \mathbf{y};$$

the column player's expected payoff is the negative of this. Thus, a *mixed Nash equilibrium* is a pair $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ such that

$$\hat{\mathbf{x}}^T A \hat{\mathbf{y}} \geq \mathbf{x}^T A \hat{\mathbf{y}} \qquad \text{for all distributions } \mathbf{x} \text{ over rows}$$

and

$$\hat{\mathbf{x}}^T A \hat{\mathbf{y}} \leq \hat{\mathbf{x}}^T A \mathbf{y} \qquad \text{for all distributions } \mathbf{y} \text{ over columns.}$$

Suppose you're due to play a zero-sum game with someone else. Would you rather move — meaning commit to a mixed strategy — first or second? Intuitively, there is only a first-mover disadvantage, since the second player can adapt to the first player's strategy. The Minimax Theorem is the amazing statement that *it doesn't matter*.

Theorem 3.1 (Minimax Theorem) *For every two-player zero-sum game A,*

$$\max_{\mathbf{x}} \left(\min_{\mathbf{y}} \mathbf{x}^T A \mathbf{y} \right) = \min_{\mathbf{y}} \left(\max_{\mathbf{x}} \mathbf{x}^T A \mathbf{y} \right). \tag{8}$$

On the left-hand side of (8), the row player moves first and the column player second. The column plays optimally given the strategy chosen by the row player, and the row player plays optimally in light of the column player's behavior. On the right-hand side of (8), the roles of the two players are reversed.

The Minimax Theorem is equivalent to the statement that every two-player zero-sum game has at least one mixed Nash equilibrium (see the Exercises). Borel, who you might know from his work developing measure-theoretic probability, was interested in the latter problem. He was discouraged after he noticed the equivalence with the Minimax Theorem, which seemed intuitively false [2, Chapter 15]. In the 1920's, von Neumann proved the Minimax Theorem using Brouwer's fixed-point theorem. Many equilibrium existence results require fixed-point theorems — more on this soon — but the Minimax Theorem can also be proved with less heavy machinery. In the 1940's, von Neumann proved the Minimax Theorem again, using arguments equivalent to strong linear programming duality.[1] This is why, when a very nervous George Dantzig first explained his new simplex algorithm to von Neumann, the latter was able to respond with an impromptu lecture outlining the corresponding duality theory [4]. These days, we don't even need linear programming per se to prove the Minimax

[1] This implies that minimax pairs and, equivalently, Nash equilibria, can be computed in polynomial time in two-player zero-sum games. See the Problems for details.

Theorem — all we need is the existence of a no-(external-)regret algorithm, such as the multiplicative weights algorithm![2]

Proof of Theorem 3.1: Since it's only worse to go first, the left-hand side of (8) is at most the right-hand side: if $\hat{\mathbf{x}}$ is optimal for the row player when it plays first, it always has the option of playing $\hat{\mathbf{x}}$ when it plays second. We turn our attention to the reverse inequality.

Given a two-player zero-sum game A, suppose both players play the game using their favorite no regret algorithms, for a long enough time T so that both have expected regret at most ϵ with respect to every fixed strategy. For example, if both players use the MW algorithm from last lecture, then $T = \Theta((\ln n)/\epsilon^2)$ is long enough.[3]

Formally, let $\mathbf{p}^1, \ldots, \mathbf{p}^T$ and $\mathbf{q}^1, \ldots, \mathbf{q}^T$ be the mixed strategies played by the row and column players, respectively, as advised by their no-regret algorithms. The inputs to the no-regret algorithms at time t are $A\mathbf{q}^t$ for the row player and $(\mathbf{p}^t)^T A$ for the column player — the expected payoff of each strategy on day t, given the mixed strategy played by the other player on day t. Set

$$\hat{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{p}^t$$

to be the time-averaged mixed-strategy of the row player,

$$\hat{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{q}^t$$

to be the time-averaged mixed-strategy of the column player, and

$$v = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{p}^t)^T A\mathbf{q}^t$$

the time-averaged expected payoff of the row player.

Adopt the row player's perspective. Since its expected regret is at most ϵ with respect to every row i and corresponding pure strategy e_i, we have

$$(e_i)^T A\hat{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T} (e_i)^T A\mathbf{q}^t \leq \frac{1}{T}\sum_{t=1}^{T} (\mathbf{p}^t)^T A\mathbf{q}^t + \epsilon = v + \epsilon. \tag{9}$$

Since an arbitrary row mixed strategy $\mathbf{x}$ is just a distribution over the e_i's, by linearity (9) implies that

$$\mathbf{x}^T A\hat{\mathbf{y}} \leq v + \epsilon \tag{10}$$

[2]It is not hard to prove that the Minimax Theorem and strong linear programming duality are equivalent, so this argument establishes the latter as well!

[3]Last lecture we defined online decision-making problems and regret in terms of cost vectors. It is straightforward to adjust the definitions for payoff vectors. It is also straightforward to adapt the MW algorithm to payoff-maximization while preserving its optimal regret bound of $O(\sqrt{(\ln n)/T})$; see the Exercises for details.

for every mixed row strategy $\mathbf{x}$.

A symmetric argument from the column player's perspective, using that its expected regret is also at most ϵ for every fixed strategy, shows that

$$\hat{\mathbf{x}}^T A\mathbf{y} \geq v - \epsilon \tag{11}$$

for every mixed column strategy $\mathbf{y}$. Thus

$$\begin{aligned}
\max_{\mathbf{x}} \left(\min_{\mathbf{y}} \mathbf{x}^T A\mathbf{y} \right) &\geq \min_{\mathbf{y}} \hat{\mathbf{x}}^T A\mathbf{y} \\
&\geq v - \epsilon && (12) \\
&\geq \max_{\mathbf{x}} \mathbf{x}^T A\hat{\mathbf{y}} - 2\epsilon && (13) \\
&\geq \min_{\mathbf{y}} \left(\max_{\mathbf{x}} \mathbf{x}^T A\mathbf{y} \right) - 2\epsilon,
\end{aligned}$$

where (12) and (13) follow from (11) and (10), respectively. Taking the limit as $\epsilon \downarrow 0$ (and $T \to \infty$) completes the proof. ■

There are a number of easy but useful corollaries of the Minimax Theorem and its proof. First, in the limit, the mixed strategies $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are a Nash equilibrium of the game A. This establishes the existence of Nash equilibria in all two-player zero-sum games. This is remarkable because most equilibrium existence results require the use of a fixed-point theorem. Second, the equivalence between Nash equilibria and minimax pairs — row and column mixed strategies $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ that optimize the left- and right-hand sides of (8), respectively — implies a "mix and match" property: if $(\mathbf{x}^1, \mathbf{y}^1)$ and $(\mathbf{x}^2, \mathbf{y}^2)$ are Nash equilibria of the same two-player zero-sum game, then so are $(\mathbf{x}^1, \mathbf{y}^2)$ and $(\mathbf{x}^2, \mathbf{y}^1)$.

References

[1] A. Blum and Y. Mansour. From external to internal regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

[2] V. Chvátal. *Linear Programming*. Freeman, 1983.

[3] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 2012. Fourth edition.

[4] J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver, editors. *History of Mathematical Programming: A Collection of Personal Reminiscences*. CWI, 1991.

CS364A: Algorithmic Game Theory
Lecture #19: Pure Nash Equilibria and PLS-Completeness*

Tim Roughgarden†

December 2, 2013

1 The Big Picture

We now have an impressive list of tractability results — polynomial-time algorithms and quickly converging learning dynamics — for several equilibrium concepts in several classes of games. Such tractability results, especially via reasonably natural learning processes, lend credibility to the predictive power of these equilibrium concepts. See also Figure 1.

> [Lecture 17] In general games, no-(external)-regret dynamics converges quickly to an approximate coarse correlated equilibrium (CCE).

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

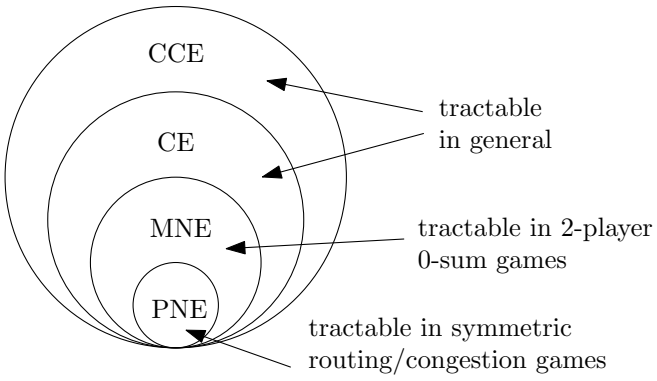


Figure 1: The hierarchy of solution concepts.

[Lecture 18] In general games, no-swap-regret dynamics converges quickly to an approximate correlated equilibrium (CE).

[Lecture 18] In two-player zero-sum games, no-(external)-regret dynamics converges quickly to an approximate mixed Nash equilibrium (MNE).

[Lecture 16] In atomic routing games that are symmetric — that is, all players share the same source and sink — ϵ-best-response dynamics converges quickly to an approximate pure Nash equilibrium (PNE).

Also, Problem 32 shows how to use linear programming to compute an exact CCE or CE of a general game, or an exact MNE of a two-player zero-sum game. Problem 22 shows how to use minimum-cost flow to compute an exact PNE of a symmetric atomic selfish routing game.

While the list above is not exhaustive, it does cover a significant chunk of the general results known for efficiently learning and computing equilibria. We'd like more, of course, such as tractability results for MNE in (non-zero-sum) two-player games and PNE for (non-symmetric) atomic routing games. No such results are known, despite non-trivial effort by many smart people. Do we merely need a new and clever idea, or are these problems intrinsically difficult? How might we prove limitations on what can be computed or learned efficiently?

These questions are in the wheelhouse of computational complexity theory. Why is it so easy to come up with computationally efficient algorithms for the minimum-spanning tree problem and so difficult to come up with one for the Travelling Salesman problem? Could it be that no efficient algorithm for the latter problem exists? If so, how can we prove it? If we can't prove it, how can we nevertheless amass evidence of intractability? These questions are, of course, addressed by the theory of NP-completeness. This lecture and the next describe analogs of NP-completeness for equilibrium computation problems. We address the technically simpler case of PNE in routing and congestion games first, and discuss MNE of bimatrix games next lecture.

2 Local Search Problems

When Johnson, Papadimitriou, and Yannakakis [2] initiated the complexity-theoretic study of local search problems, they did not have equilibrium computation in mind. However, the theory they developed is perfectly suited for reasoning about PNE computation in routing and congestion games, as we'll see in Section 3. To foreshadow the connection, computing a PNE of a congestion game is equivalent to computing a local minima of the Rosenthal potential function (Lecture 13). Hence, complexity theory for local search problems is immediately relevant to these equilibrium computation problems.

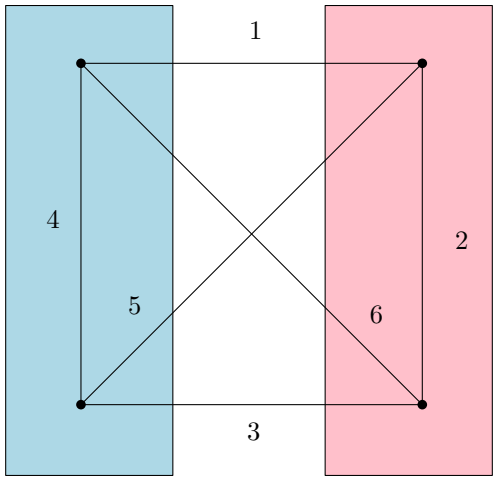


Figure 2: Max-cut instance and a local optimum that is not a global optimum.

2.1 Canonical Example: The Maximum Cut Problem

A canonical problem through which to study local search is the *maximum cut* problem. The input is an undirected graph $G = (V, E)$ with a nonnegative weight $w_e \geq 0$ for each edge. Feasible solutions correspond to cuts $(S, \bar{S})$, where $(S, \bar{S})$ is a partition of V into two non-empty sets. The objective function that we wish to maximize is the total weight of the cut edges — the edges with one endpoint in each of $S, \bar{S}$. Unlike the minimum cut problem, the maximum cut problem is NP-hard.

Local search is a natural heuristic that is useful for many NP-hard problems, including the maximum cut problem. The algorithm is very simple:

1. Start with an arbitrary cut $(S, \bar{S})$.

2. While there is an improving local move, make one.

By a *local move*, we mean moving a vertex v from one side of the cut to the other (keeping both sides non-empty). For example, when moving a vertex v from S to $\bar{S}$, the increase in objective function value is

$$\underbrace{\sum_{u \in S} w_{uv}}_{\text{newly cut}} - \underbrace{\sum_{u \in \bar{S}} w_{uv}}_{\text{newly uncut}};$$

if this difference is positive, then this is an improving local move. Local search terminates at a solution with no improving move, a *local optimum*. A local optimum need not be a global optimum; see Figure 2.

Since there are more local optima than global optima, they are generally easier to find. For example, consider the special case of maximum cut instances in which every edge has unit weight. Computing a global maximum remains an NP-hard problem, but computing a local maximum is easy. Because the objective function in this case can only take on the values $\{0, 1, 2, \ldots, |E|\}$, local search terminates (at a local maximum) in at most $|E|$ iterations.

There is no known polynomial-time algorithm (local search or otherwise) for computing local optima of maximum cut instances with general nonnegative weights. How might we amass evidence that no such algorithm exists?

The strongest type of negative result would be an unconditional one: a proof that there is no polynomial-time algorithm for the problem. No one knows how to prove unconditional results like this; in particular, such a result would separate P from NP. Instead, a natural response is to try to prove that finding local maxima of maximum cut instances is an NP-complete problem. In the next lecture we'll see why this is also too strong a negative result to shoot for. Instead, we'll develop an analog of NP-completeness tailored for local search problems.

2.2 PLS: Abstract Local Search Problems

This section and the next make precise the idea that the problem of computing a local optimum of a maximum cut instance is *as hard as any other local search problem*. This statement is in the spirit of an NP-completeness result, which establishes a problem to be as hard as any problem with efficiently verifiable solutions. For such "hardest" local search problems, we don't expect there to be any clever, problem-dependent algorithm that improves significantly over local search. This parallels the idea that for NP-complete problems, we don't expect there to be an algorithm that improves significantly over brute-force search. A byproduct of the theory developed in this and the next section is exponential lower bounds on the number of iterations required by local search to reach a local optimum.

What could we mean by "any other local search problem?" One interpretation of the definition of NP problems is that an efficient verifier of purported solutions is in some sense the minimal ingredient necessary to execute brute-force search through the set of candidate solutions. So what are the minimal ingredients necessary to run local search?

An abstract local search problem is specified by three polynomial-time algorithms.

1. The first algorithm takes as input an instance and outputs an arbitrary feasible solution. [In MaxCut, it outputs an arbitrary cut.]

2. The second algorithm takes as input an instance and a feasible solution, and returns the objective function value of the solution. [In MaxCut, it outputs the total weight of the edges crossing the provided cut.]

3. The third algorithm takes as input an instance and a feasible solution and either reports "locally optimal" or produces a better solution. [In MaxCut, it checks all $|V|$ local moves. If none are improving it outputs "locally optimal;" otherwise, it executes an improving local move and outputs the resulting cut.][1]

[1] There are some details we're glossing over. For example, all algorithms should check if the given instance is legitimate. There is also some canonical interpretation when an algorithm misbehaves, by running too long or outputting something invalid. For example, we can interpret the output of the third algorithm as "locally optimal" unless is outputs a feasible solution better than the previous one (as verified by the second algorithm) within a specified polynomial number of steps.

Every problem in PLS admits a local search algorithm: given an instance, use the first algorithm to obtain a start state, and iteratively apply the third algorithm until a locally optimal solution is reached. Since the objective function values of the candidate solutions strictly decrease until a locally optimal solution is found, and since there are only finitely many distinct candidate solutions, this procedure eventually terminates.[2] Because there can be an exponential number of feasible solutions (e.g., in a maximum cut instance), this local search procedure need not run in polynomial time.

The goal in an abstract local search problem is to compute a local optimum. This can be done by running the local search algorithm, but any correct algorithm for computing a local optimum is also allowed. The complexity class PLS is, by definition, the set of all such abstract local search problems [2]. Most if not all of the local search problems you've ever seen can be cast as PLS problems.

2.3 PLS-Completeness

Our goal is to prove that the problem of computing a local optimum of a maximum cut instance is as hard as any other local search problem. Having formalized "any other search problem," we now formalize the phrase "as hard as." This done using reductions, which are familiar from the theory of NP-completeness.

Formally, a *reduction* from a problem $L_1 \in PLS$ to a problem $L_2 \in PLS$ is two polynomial-time algorithms:

1. Algorithm A maps every instance $x \in L_1$ to an instance $A(x) \in L_2$.

2. Algorithm B maps every local optimum of $A(x)$ to a local optimum of x.

The definition of a reduction ensures that if we can solve the problem L_2 in polynomial time then, by combining it with algorithms A and B in the reduction in the natural way, we can also solve the problem L_1 in polynomial time.

Definition 2.1 ([2]) A problem $L \in PLS$ is *PLS-complete* if every problem of PLS reduces to it.

By definition, there is no polynomial-time algorithm for computing a local optimum of a PLS-complete problem, unless $PLS \subseteq P$. Most researchers believe that $PLS \not\subseteq P$, though confidence is not as strong as for the $P \neq NP$ conjecture.

A PLS-complete problem is a *single* local search problem that simultaneously encodes *every* local search problem. If we didn't already have the remarkable theory of NP-completeness to guide us, we might not believe that a PLS-complete problem could exist. As with NP-complete problems, though, PLS-complete problems *do* exist. Even more remarkably, many concrete problems that we care about are PLS-complete [2, 3]. In particular:

[2]Since each of the three algorithms runs in time polynomial in the input size, we are implicitly forcing feasible solutions to have polynomial description length. Hence, there are at most exponentially many feasible solutions.

Fact 2.2 ([3]) *Computing a local maximum of a maximum cut instance with general non-negative weights is a PLS-complete problem.*

The first step in developing a theory of PLS-completeness is to prove an analog of Cook's theorem, meaning an initial complete problem. Cook's Theorem states that 3SAT is an NP-complete problem; Johnson et al. [2] proved that a problem concerning Boolean circuits called "CircuitFlip" is PLS-complete. Once a first complete problem is identified, more can be obtained via reductions. Some such reductions were given in [2] and many more, including to the maximum cut problem, were given in [3].

We already mentioned the conditional result that if $PLS \not\subseteq P$, then there is no polynomial-time algorithm (local search or otherwise) for any PLS-complete problem. Even in the unlikely event that $PLS \subseteq P$, the specific algorithm of local search requires exponential worst-case time for all known PLS-complete problems. The reason is that Johnson et al. [2] gave an unconditional exponential lower bound for local search for the CircuitFlip algorithm, and all of the reductions that have been used to establish the PLS-completeness of other problems preserve this lower bound. In particular, local search can require an exponential number of iterations to converge to general maximum cut instances.

3 Congestion Games

We mentioned *congestion games* in passing in Lecture 13 as a natural generalization of atomic selfish routing games in which strategies are abstract subsets of a ground set, rather than paths in a graph. That is, a congestion game is described by a set E of resources (previously edges), an explicitly described strategy set $\mathcal{S}_i \subseteq 2^E$ for each player $i =, 1, 2, \ldots, k$ (previously s_i-t_i paths), and a cost $c_e(i)$ for each resource $e \in E$ and possible load $i \in \{1, 2, \ldots, k\}$. All the results we proved for atomic selfish routing games — the price of anarchy bound of $\frac{5}{2}$ for affine cost functions, the existence of PNE with arbitrary cost functions, and the convergence of ϵ-best-response dynamics to an approximate PNE in symmetric games with α-bounded jump cost functions — hold more generally, with exactly the same proofs, in congestion games.

We claim that the problem of computing a PNE of a congestion game — any PNE, if there are many — is a PLS problem. Essentially, the claim follows from the correspondence between best-response dynamics in a congestion game and local search with respect to the Rosenthal potential function (Lecture 13). Proving the claim formally involves describing the three algorithms that define a PLS problem. The first algorithm takes as input a congestion game, as described above, and returns an arbitrary strategy profile — for example, where each player takes its first strategy. The second algorithm takes a congestion game and a strategy profile $\mathbf{s}$, and returns the value of the Rosenthal potential function

$$\Phi(\mathbf{s}) = \sum_{e \in E} \sum_{i=1}^{n_e(\mathbf{s})} c_e(i), \tag{1}$$

where $n_e(\mathbf{s})$ is the number of players in the given profile $\mathbf{s}$ that use a strategy that includes

resource e, and c_e is the given cost function of e. The third algorithm checks whether or not the given strategy profile is a PNE; if so, it reports "locally optimal;" if not, it executes an iteration of best-response dynamics and returns the resulting outcome (which has smaller potential function value). This can be done in time polynomial in the description of the given congestion game.

More interesting is that computing a PNE of a congestion game is as hard as every other local search problem.

Theorem 3.1 ([1]) *The problem of computing a PNE of a congestion game is PLS-complete.*

Of course, problems that require computing a PNE with additional properties, like the best or worst PNE, can only be harder.

Proof: We give a reduction from the maximum cut problem, which is PLS-complete [2]. We are given a graph $G = (V, E)$ with nonnegative edge weights $\{w_e\}_{e \in E}$. The first algorithm A of the reduction constructs a congestion game as follows:

1. Players correspond to the vertices V.
2. There are two resources for each edge $e \in E$, r_e and $\bar{r}_e$.
3. Player v has two strategies, each comprising $|\delta(v)|$ resources, where $\delta(v)$ is the set of edges incident to v in G: $\{r_e\}_{e \in \delta(v)}$ and $\{\bar{r}_e\}_{e \in \delta(v)}$.
4. The cost of a resource r_e or $\bar{r}_e$ is 0 if one player uses it, and w_e if two players use it.

The two strategies of a player can be regarded as choosing which side of a cut, S or $\bar{S}$, the player is on. Note that for an edge $e = (u, v)$ of G, there are only two players that have a strategy containing r_e or $\bar{r}_e$: the players u and v. In every strategy profile, the combined load on the twin resources r_e and $\bar{r}_e$ is exactly two: either both players u, v use the same resource (corresponding to choosing the same side of a cut) or exactly one player u, v uses each of $r_e, \bar{r}_e$ (corresponding to choosing different sides of a cut). This is why we specify cost functions only for one or two players.

There is a bijection between strategy profiles of this congestion game and cuts of the given graph G (allowing also cuts that are empty on one side), with cut $(S, \bar{S})$ corresponding to the profile in which every player corresponding to $v \in S$ (respectively, $v \in \bar{S}$) chooses its strategy that contains resources of the form r_e (respectively, $\bar{r}_e$).

This bijection maps cuts of G with weight $w(S, \bar{S})$ to strategy profiles with Rosenthal potential value (1) equal to $\sum_{e \in E} w_e - w(S, \bar{S})$. To see this, fix a cut $(S, \bar{S})$. For an edge e cut by $(S, \bar{S})$, the load on the corresponding resources r_e and $\bar{r}_e$ is 0, so these resources contribute nothing to the Rosenthal potential of the strategy profile. For an edge e not cut by $(S, \bar{S})$, the load on one of the resources $r_e, \bar{r}_e$ will be 2, and the other's load will be 0. This pair of resources contributes w_e to the potential function (1). Thus, the potential function value of the corresponding strategy profile is the total weight of edges uncut by $(S, \bar{S})$, or $\sum_{e \in E} w_e - w(S, \bar{S})$.

Cuts of G with larger weight thus correspond to strategy profiles with smaller Rosenthal potential. Local maxima of G correspond to local minima of the potential function — that is, to PNE of the congestion game. Mapping PNE of the congestion game back to local optima of G is thus trivial to do, and the reduction is complete. ■

The proof of Theorem 3.1 also provides a good example of how reductions between PLS problems tend to preserve unconditional lower bounds on the running time of local search. Specifically, it establishes a bijection between local search in maximum cut instances and best-response dynamics in congestion games. Since the former process can require an exponential number of iterations to converge, so can the latter.

Remark 3.2 The theory of PLS-completeness is useful even if you care only about concrete lower bounds on specific dynamics like best-response dynamics, and not about computational complexity per se. It can be very difficult to prove such lower bounds from scratch, and is potentially much easier to simply reduce a PLS-complete problem to the equilibrium computation problem of interest.

We can build on the proof of Theorem 3.1 to extend PLS-completeness to the special case of *symmetric* congestion games.

Theorem 3.3 ([1]) *The problem of computing a PNE of a symmetric congestion game is PLS-complete.*

At first blush, Theorem 3.3 might seem to contradict positive results that we've already proved. First, recall the polynomial-time convergence result from Lecture 16 for ϵ-best-response dynamics. We proved that convergence result only for atomic selfish routing games with a common source and sink, but the same result and proof extend without change to symmetric congestion games. Thus, Theorem 3.3 implies that there is a big difference between exact and approximate PNE, and between exact and approximate best-response dynamics in symmetric congestion games.[3] Approximate PNE are tractable and approximate best-response dynamics converge quickly, while exact PNE are intractable and exact best-response dynamics can require an exponential number of iterations to converge.

Second, recall from Problem 22 that a PNE of a symmetric atomic selfish routing game can be computed in polynomial time using minimum-cost flow. Theorem 3.3 implies that this positive result cannot be extended to abstract symmetric congestion games (unless $PLS \subseteq P$).[4]

Proof of Theorem 3.3: We reduce the problem of computing a PNE of an asymmetric congestion game — PLS-complete by Theorem 3.1 — to that of computing a PNE of a symmetric

[3] Our proof of Theorem 3.3 will also violate the α-bounded jump assumption we made in Lecture 16, but the reduction can modified to respect this condition.

[4] In the asymmetric case, the complexity of computing a PNE is PLS-complete in both atomic selfish routing games and more general abstract congestion games [1]. Also, in the asymmetric case, computing even an approximate PNE is PLS-complete, and ϵ-best-response dynamics can require an exponential number of iterations to converge [4].

congestion game. Given an asymmetric congestion game with k players and arbitrary strategy sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$, construct a "symmetrized version" as follows. The player set remains the same. The old resource set is augmented by k additional resources $r_1, \ldots, r_k$. The cost function of each of these is defined to be zero if used by only one player, and extremely large if used by two or more. Each strategy of $\mathcal{S}_i$ is supplemented by the resource r_i, and any player can adopt any one of these augmented strategies. The key insight is that at a pure Nash equilibrium of the constructed symmetric game, each player adopts the identity of exactly one player from the original (asymmetric) game. This follows from the large penalty incurred by two players that choose strategies that share one of the new resources. Such a pure Nash equilibrium is then easily mapped to one of the original asymmetric game. ■

As an exercise, why doesn't the reduction in Theorem 3.3 work for atomic routing games? The obvious way to symmetrize an asymmetric routing game with sources $s_1, \ldots, s_k$ and sinks $t_1, \ldots, t_k$ is to add new source and sink vertices s and t, and new arcs $(s, s_1), \ldots, (s, s_k)$ and $(t_1, t), \ldots, (t_k, t)$, each with a cost function that is zero with one player and extremely large with two or more players.

References

[1] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[2] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[3] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991.

[4] Alexander Skopalik and Berthold Vcking. Inapproximability of pure nash equilibria. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2008.

CS364A: Algorithmic Game Theory
Lecture #20: Mixed Nash Equilibria and PPAD-Completeness*

Tim Roughgarden†

December 4, 2013

Today we continue our study of the limitations of learning dynamics and polynomial-time algorithms for converging to and computing equilibria. Recall that we have sweeping positive results for coarse correlated and correlated equilibria, which are tractable in arbitrary games. We have only partial positive results for pure Nash equilibria of routing and congestion games, and last lecture we developed the theory of PLS-completeness to explain our limited success. In this lecture we focus on mixed Nash equilibria (MNE). Our positive results so far have been limited to the special case of two-player, zero-sum games (Lecture 18). This lecture develops theory that suggests that there cannot be significantly more general positive results.

1 The Problem: Computing a MNE of a Bimatrix Game

Formally, we study the problem of computing a MNE of a bimatrix game.[1] The input is two $m \times n$ payoff matrices A and B — one for the row player, one for the column player. In zero-sum games, $B = -A$. The goal is to compute mixed strategies $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ such that

$$\hat{\mathbf{x}}^T A \hat{\mathbf{y}} \geq \mathbf{x}^T A \hat{\mathbf{y}} \tag{1}$$

for all row mixed strategies $\mathbf{x}$ and

$$\hat{\mathbf{x}}^T B \hat{\mathbf{y}} \geq \hat{\mathbf{x}}^T B \mathbf{y} \tag{2}$$

for all column mixed strategies $\mathbf{y}$.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1] An alternative generalization of two-player zero-sum games is three-player zero-sum games. Such games include two-player non-zero-sum games as a special case — do you see why?

There is no known polynomial-time algorithm for computing a MNE of a bimatrix game, and many smart people of tried to come up with one. This lecture develops the relevant complexity theory for arguing that the problem may be inherently intractable. The goal is to prove that the problem is complete for a suitable complexity class. But which class? This is a tricky question, and it took years for the leading thinkers in the field to figure out the answer. Before we explain the solution, let's understand why plain old NP-completeness is not the right intractability notion for equilibrium computation. The discussion in the next section also applies to the PLS problems discussed last lecture, thereby justifying our development of PLS-completeness as a weaker analog of NP-completeness.

2 NP Search Problems (FNP)

NP problems are decision problems, where the correct answer to an instance is either "yes" or "no". Equilibrium computation problems are not decision problems; the output should be a bona fide equilibrium. To address this typechecking error, we work with the closely related class FNP, for "functional NP." FNP problems are just like NP problems except that, for "yes" instances, we demand that a solution be produced. These are also call *search* problems.

More formally, an algorithm for an FNP problem takes as input an instance of an NP problem, like a SAT formula or an undirected graph. The responsibility of the algorithm is to output a solution, or "witness," like a satisfying assignment or a Hamiltonian cycle, provided one exists. If there is no solution, the algorithm should output "no." Reductions between search problems are defined as in the last lecture via two polynomial-time algorithms, the first algorithm A mapping instances x of one problem to instances $A(x)$ of another, the second algorithm B mapping solutions of $A(x)$ to solutions to x (and "no" to "no").

Your intuition for NP works fine for FNP. For example, the functional version of SAT is an FNP-complete problem. The proof of Cook's theorem — which effectively constructs the algorithm A above from an arbitrary NP problem to SAT — establishes a bijective correspondence between the witnesses of the given NP verifier for the given instance and the satisfying assignments of the constructed SAT formula. The algorithm B is a straightforward implementation of this correspondence.

The class PLS of local search problems, defined last lecture, is a subset of FNP. The witnesses of a PLS problem are its local optima, and the third algorithm in the PLS problem description acts as an efficient verifier of witnesses. In fact, the third algorithm of a PLS problem does considerably more than is asked of an NP verifier — in addition to certifying local optima, when a solution is not locally optimal, the PLS algorithm does not merely say "no," it offers an alternative solution with superior objective function value.

The problem of computing a MNE of a bimatrix game also belongs to FNP. This assertion amounts to proving that, given mixed strategies $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ in a bimatrix game (A, B), it can be checked efficiently whether or not $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ constitute an MNE of the game. This is not entirely obvious, since the equilibrium conditions (1) and (2) reference an infinite number of mixed strategies. Fortunately, it is enough to check only pure-strategy deviations.

Equivalently, a mixed strategy $\hat{\mathbf{x}}$ by the row player is a best response to a column strategy $\hat{\mathbf{y}}$ if and only if $\hat{\mathbf{x}}$ randomizes only over rows with maximum expected payoff (w.r.t. $\hat{\mathbf{y}}$); see also the Exercises.[2]

3 NP Search Problems with Guaranteed Witnesses ($TFNP$)

Could computing a MNE of bimatrix game be FNP-complete? Being as hard as every problem in FNP would constitute strong evidence of intractability. Intriguingly, FNP-completeness would have astonishing consequences.

Theorem 3.1 ([8]) *If the problem of computing a MNE of bimatrix game is FNP-complete, then $NP = coNP$.*

While $NP = coNP$ doesn't directly imply $P = NP$, it is thought to be an equally unlikely state of affairs. For example, if $NP = coNP$, then there are short, efficiently verifiable proofs for the $coNP$-complete UNSAT problem. Convincing someone that a SAT formula is satisfiable is easy enough — just exhibit a satisfying assignment — but how would you quickly convince someone that none of the exponentially many truth assignments are satisfying? Most researchers believe that there is no way to do it — that $NP \neq coNP$. If this is indeed the case, then Theorem 3.1 implies that the problem of computing a MNE of a bimatrix game is not FNP-complete.

Proof of Theorem 3.1: The proof is a bit of a mind-binder, but it is not long. Assume there is a reduction, in the same sense of the PLS reductions described last lecture, from the functional SAT problem to the problem of computing a MNE of a bimatrix game. By the definition of a reduction, there exist the following two algorithms:

1. A polynomial-time algorithm A that maps every SAT formula φ to a bimatrix game $A(\varphi)$.

2. A polynomial-time algorithm B that maps every MNE $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ of a game $A(\varphi)$ to:

   (a) a satisfying assignment $B(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ of $A(\varphi)$, if one exists;

   (b) the string "no," otherwise.

We claim that the existence of these algorithms A and B imply that $NP = coNP$. Indeed, consider an arbitrary "no" instance φ of SAT, and an arbitrary MNE $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ of the game $A(\varphi)$.[3] We claim that $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a short, efficiently verifiable proof of the unsatisfiability

[2] To be completely rigorous, we also need to argue that there are MNE whose description (in bits) has length polynomial in the input size. This follows from the observation that, given the supports of a MNE — the strategies played with nonzero probability — suitable mixing probabilities can be recovered by solving a linear system of equations. See the Problems for more details.

[3] Crucially, Nash's Theorem ensures that $A(\varphi)$ has at least one MNE. Also, as noted above, every bimatrix game has an MNE whose description length is polynomial in that of the game. We discuss the proof of Nash's Theorem in Section 7.

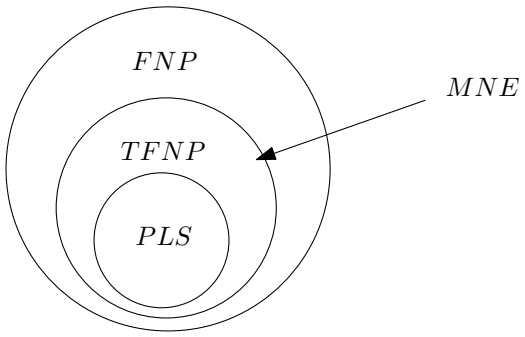


Figure 1: MNE are in the TFNP complexity class.

of φ; this implies $NP = coNP$. Indeed, given an alleged unsatisfiability certificate $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ for φ, one performs two checks: (i) compute the game $A(\varphi)$ using algorithm A and verify that $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a MNE of $A(\varphi)$; (ii) use the algorithm B to verify that $B(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is the string "no." If $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ passes both of these tests, then correctness of the algorithms A and B implies that φ is indeed unsatisfiable. ■

What's really going on in the proof of Theorem 3.1 is a mismatch between an FNP-complete problem like $FSAT$, where an instance may or may not have a witness, and a problem like computing an MNE, where Nash's theorem guarantees at least one (polynomial-length) witness in every instance. While the correct answer to an instance of FSAT might well be "no;" a correct answer to an instance of computing a MNE of a game is always a witness (a MNE). The subset of FNP problems for which every instance has at least one witness is called $TFNP$, for "total functional NP." The proof of Theorem 3.1 shows more generally that if *any* $TFNP$ problem is FNP-complete, then $NP = coNP$. In particular, since every instance of a PLS problem has at least one witness — local search has to stop somewhere, necessarily at a local optimum — PLS is a subset of $TFNP$. Hence no PLS problem, such as computing a PNE of a routing or congestion game, can be FNP-complete unless $NP = coNP$. See Figure 1. This justifies last lecture's development of PLS-completeness, a weaker analog of FNP-completeness tailored for local search problems.

4 Complete Problems: Syntactic vs. Semantic Complexity Classes

Membership in $TFNP$ precludes proving that computing a MNE of a bimatrix game is FNP-complete. The sensible refined goal, then, is to prove that the problem is $TFNP$-complete — as hard as any other problem in $TFNP$. Unfortunately, $TFNP$-completeness is also too ambitious a goal. The reason is that $TFNP$ does not seem to have complete problems. To explain, think about the complexity classes you know about that *do* have complete problems — NP of course, and also P and $PSPACE$. What do these complexity classes have in common? They are "syntactic," meaning that membership can be characterized via

acceptance by some concrete computational model — polynomial-time or polynomial-space Turing machines, for example. In this sense, there is a "generic reason for membership" in these complexity classes.

$TFNP$ has no obvious generic reason for membership, and as such is called a "semantic class."[4] For example, the problem of computing a MNE of a bimatrix game belongs to $TFNP$ because of Nash's theorem guaranteeing the existence of a MNE — essentially, for topological reasons (see Section 7). Another problem in $TFNP$ is factoring — given a positive integer, output its factorization. Here, membership in $TFNP$ has an algebraic or number-theoretic explanation. Can the existence of a MNE and of an integer factorization be regarded as separate instantiations of some "generic" $TFNP$ argument? No one knows the answer.

5 $PPAD$: A Syntactic Subclass of $TFNP$

Papadimitriou [11] proposed a path to progress: identify *subclasses* of $TFNP$ such that:

1. The class contains interesting computational problems not known to be in P (like computing a MNE).

2. The class has complete problems; roughly equivalently, the class has a "syntactic" definition in the form of a generic reason for membership.

We have already seen an example of such a subclass: the complexity class PLS from last lecture. For example, computing a local optimum of a maximum cut instance is a PLS-complete problem that is not known to be polynomial-time solvable. Our definition of PLS is syntactic in that PLS problems are precisely those defined by a particular computational model, as induced by the three polynomial-time algorithms that define such a problem. The generic reason for membership in PLS (and hence $TFNP$) is that local search, as defined by these three given algorithms, is guaranteed to eventually terminate at a local optimum.

The right complexity class to study the computation of MNE in bimatrix games is called $PPAD$.[5] We describe $PPAD$ primarily via analogy with the class PLS, which is similar in spirit but different in details.

We can think of a local search problem as the problem of computing a sink vertex of a directed acyclic graph (Figure 2), where nodes represent feasible solutions and arcs represent improving local moves. Note that the number of nodes in this directed graph is generally exponential in the size of a problem instance — for example, in a maximum cut instance, nodes correspond to cuts of the original graph. The three algorithms that define a PLS problem enable a straightforward path-following algorithm in this directed graph — the first algorithm gives a starting node and third algorithm provides the next arc to traverse from a non-sink node.

[4] There are many other interesting examples, such as $NP \cap coNP$.

[5] Allegedly standing for "polynomial parity argument, directed version" [11].

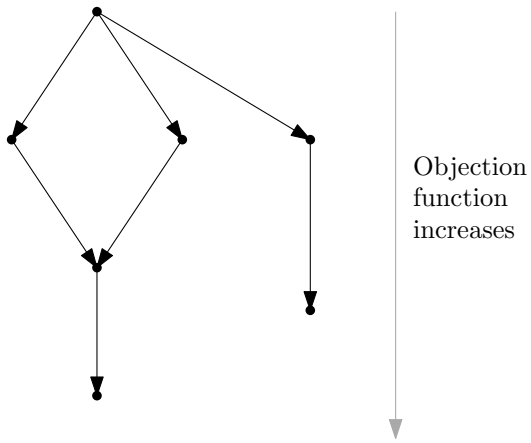


Figure 2: A PLS problem is characterized by a DAG with an objective function.

$PPAD$ problems are those solvable by a particular class of naive directed path-following algorithms — this is the generic reason for membership. In this sense, $PPAD$ problems are similar to PLS problems, which are solvable by following a path through a directed acyclic graph. $PPAD$ problems can be thought of as directed graphs (Figure 3), where nodes again correspond to "intermediate solutions," , and arcs to "next moves." Like with PLS problems, the number of nodes in generally exponential in the parameters of interest. Rather than being directed acyclic, a $PPAD$ directed graph has in- and out-degree at most 1, and at least one source node. Unlike a PLS problem, there is no objective function, and a $PPAD$ directed graph can possess cycles. Like a PLS problem, a $PPAD$ problem is formally specified by three algorithms — this is a "syntactic definition." The first algorithm suggests an initial solution that is a source vertex. The third algorithm either verifies that the current intermediate solution is a witness — by definition, a source or sink vertex different from the starting point — or else returns the next intermediate solution. Naive path-following — invoking the first algorithm once and the third algorithm until the other end of the path is reached — is guaranteed to terminate at a witness.[6] This path might have exponential length, however, so the non-trivial question is whether or not a witness of a $PPAD$ problem can be computed in polynomial time.

What do $PPAD$ problems have to do with computing MNE, or anything else for that matter? To get a feel for this complexity class, we discuss next a canonical example of a $PPAD$ problem.

[6]The primary role of the second algorithm is to keep the third algorithm honest — this is technically necessary so that naive path-following is guaranteed to terminate, no matter what the given three algorithms are. Recall that in a PLS problem, the second algorithm, which computes the objective function value, can be used to verify that the third algorithm suggests a superior solution. If the third algorithm fails to produce a better solution, we just interpret its output as "locally optimal." In a $PPAD$ problem, the second algorithm enforces that the corresponding directed graph has in-degree at most 1. It takes as input an intermediate solution and outputs the previous one. If the third algorithm declares that y should follow x, but the second algorithm does not agree that x should precede y, then we interpret x as a sink (and hence a witness).

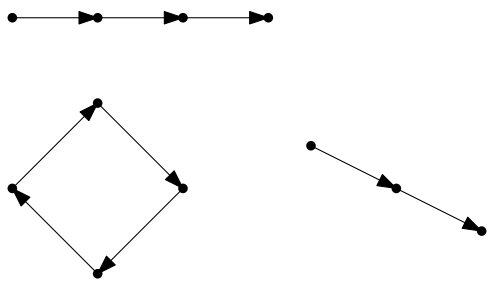


Figure 3: PPAD problems are characterized by a graph where each node has in- and out-degree at most 1.

6 A Canonical $PPAD$ Problem: Sperner's Lemma

This section presents a computational problem that is clearly well matched with the $PPAD$ complexity class. The next section describes its relevance to computing a MNE of a bimatrix game.

Consider a subdivided simplex in the plane; see Figure 4. A *legal coloring* of its vertices colors the top vertex red, the left vertex green, and the right vertex blue. A vertex on the boundary must have one of the two colors of the endpoints of its side. Internal vertices are allowed to possess any of the three colors. A triangle is *trichromatic* if all three colors are represented at its corners.

Sperner's Lemma asserts that for every legal coloring, there is an odd number of trichromatic triangles (and hence at least one). The proof is constructive. Define a graph G that has one node per triangle, plus a source node outside the simplex. The graph G has one edge for each red-green side of a triangle. See Figure 4. Every trichromatic triangle corresponds to a degree-one node of G. Every triangle with one green and two red corners or two green and one red corner corresponds to a node with degree two in G. The source node of G has degree equal to the number of red-green edges on the left side of the simplex, which is an odd number. Because every graph has an even number of nodes of odd degree, G has an odd number of trichromatic triangles, which proves Sperner's Lemma.[7] Moreover, starting from the source vertex, naive path-following is guaranteed to terminate at a trichromatic triangle. Thus, computing a trichromatic triangle of a legally colored subdivided simplex is a $PPAD$ problem.[8]

[7]The same result and proof extend by induction to higher dimensions — every subdivided simplex in $\mathcal{R}^n$ with vertices legally colored with $n+1$ colors has an odd number of panchromatic subsimplices, with a different color on each corner.

[8]We are omitting some details. The graph of a $PPAD$ problem is directed, while the graph G we defined here is undirected. There is, however, a canonical way to direct the edges of this graph G.

Also, the source node of a $PPAD$ problem is supposed to have out-degree 1, while that of the graph G above has some odd degree $2k+1$. This can be corrected by splitting the source node of G into $k+1$ nodes, a source with out-degree 1 and k nodes with in- and out-degree 1.

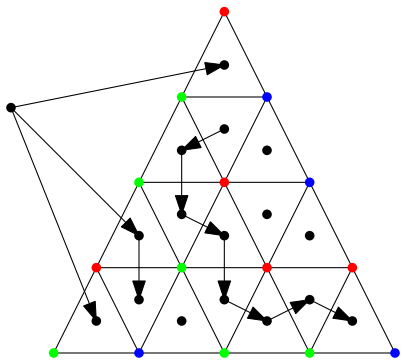


Figure 4: Every triangle with an end point is a trichromatic triangle.

7 MNE and $PPAD$

What does computing a MNE have to do with $PPAD$, the class of FNP problems solvable by a particular form of directed path-following? There are two fundamental connections.

First, Sperner's Lemma turns out to be the combinatorial heart of Nash's proof that every finite game has at least one MNE, and its proof yields a path-following algorithm for computing an approximate MNE (in exponential time). The reduction of Nash's theorem to Sperner's Lemma has two parts. The first part is use Sperner's Lemma to prove Brouwer's fixed-point theorem. Brouwer's fixed-point theorem states that every continuous function f that maps a compact convex subset C of $\mathcal{R}^n$ to itself has at least one fixed point: a point $x \in C$ with $f(x) = x$. All of the hypotheses — that f is continuous, C is bounded, C is closed, and C is convex — are necessary (see the Exercises).

Suppose we want to prove Brouwer's fixed-point theorem when C is a simplex in $\mathcal{R}^2$. Let $f : C \to C$ be continuous. Subdivide C into small triangles. Color a triangle corner green if $f(x)$ is farther from the left corner of C than x; red if $f(x)$ is farther from the top corner of C than x; and blue if x is farther from the right corner of C than x. If two of these conditions apply to x, either corresponding color can be used. (If none of them apply, x is a fixed point and there's nothing left to prove.) This results in a legal coloring of the subdivision. By Sperner's Lemma, there is at least one trichromatic triangle — a triangle whose corners are being pulled in different directions. Taking a sequence of finer and finer subdivisions, we get a sequence of ever-smaller trichromatic triangles. Because C is compact, the centers of these triangles contain a sequence $x^1, x^2, \ldots$, that converges to a point x^* in C. Because f is continuous, in the limit, $f(x^*)$ is at least as far from each of the three corners of C as x^*. This means that x^* is a fixed point of f.[9]

Nash [10] gave an elegant reduction from the existence of MNE in finite games to

[9]Here's the idea for extending the fixed-point theorem to all convex compact subsets of $\mathcal{R}^n$. First, since Sperner's Lemma extends to higher dimensions, so does Brouwer's fixed-point theorem for the special case of simplices (by the same argument). Second, every pair C_1, C_2 of compact convex subsets of equal dimension are homeomorphic — that is, there is a bijection $f : C_1 \to C_2$ with f and f^{-1} continuous — and homeomorphisms preserve fixed-point theorems (see the Exercises).

Brouwer's fixed point theorem. Here is a sketch. Consider a k-player game with strategy sets $S_1, \ldots, S_k$ and payoff functions $\pi_1, \ldots, \pi_k$. The relevant compact convex set is $C = \Delta_1 \times \cdots \times \Delta_k$, where Δ_i is the simplex representing the mixed strategies of player i. We want to define a continuous function $f : C \to C$ — from mixed strategy profiles to mixed strategy profiles — such that fixed-points of f are MNE of the given game. We define f separately for each component $f_i : C \to \Delta_i$. A natural first idea is to set f_i to be a best response of player i to the mixed strategy profiles of the other players. This does not lead to a continuous, or even well defined, function, so instead we use a "regularized" version. We set

$$f_i(x_i, \mathbf{x}_{-i}) = \operatorname*{argmax}_{x_i' \in \Delta_i} g_i(x_i', \mathbf{x}),$$

where

$$g_i(x_i, \mathbf{x}) = \underbrace{\mathbf{E}_{s_i \sim x_i', \mathbf{s}_{-i} \sim \mathbf{x}_{-i}}[\pi(\mathbf{s})]}_{\text{linear in } x_i'} - \underbrace{\|x_i' - x_i\|_2^2}_{\text{strictly convex}}.$$

The first term of the function g_i encourages a best response while the second "penalty term" discourages big changes to i's mixed strategy. Because the function g_i is strictly concave in x_i', f_i is well defined. The function $f = (f_1, \ldots, f_k)$ is continuous (see Exercises). It should be clear that every MNE of the given game is a fixed point of f. For the converse, suppose that $\mathbf{x}$ is not a MNE, with player i able to increase its expected payoff by deviating from x_i and x_i'. A simple computation shows that, for sufficiently small $\epsilon > 0$, $g_i((1-\epsilon)x_i + \epsilon x_i', \mathbf{x}) > g_i(x_i, \mathbf{x})$, and hence $\mathbf{x}$ is not a fixed point of f (see the Exercises).

There is also a second way to prove that computing a MNE of a bimatrix game is a $PPAD$ problem, via the Lemke-Howson algorithm (see [15]). The Lemke-Howson algorithm reduces computing a MNE of a bimatrix game to a path-following problem, much in the way that the simplex algorithm reduces computing an optimal solution of a linear program to following a path of improving edges along the boundary of the feasible region. The biggest difference between the Lemke-Howson algorithm and the simplex method is that the former is not guided by an objective function; all known proofs of its inevitable convergence use parity arguments akin to the one in the proof of Sperner's Lemma, thereby showing that the problem lies in $PPAD$.

The two connections between $PPAD$ and computing a MNE are incomparable. The Lemke-Howson argument applies only to games with two players, but it shows that the problem of computing an *exact* MNE of a bimatrix game belongs to $PPAD$. The path-following algorithm derived from Sperner's Lemma applies to games with any finite number of players, but only shows that the problem of computing an *approximate* MNE is in $PPAD$. In fact, with 3 or more players, the problem of computing an exact MNE of a game appears to be strictly harder than $PPAD$ problems [5].

The upshot of all this is that $PPAD$ is a subclass of $TFNP$ that contains the problem of computing a MNE of a bimatrix game. Moreover, $PPAD$ is syntactically defined, with the generic reason for membership being solvability by a naive path-following argument from a source in a directed graph with all in- and out-degrees at most 1. As such, we expect the class to admit complete problems. In fact, we have finally identified the right complexity

class for building evidence that the problem of computing a MNE of a bimatrix game is computationally intractable: it is a $PPAD$-complete problem.

Theorem 7.1 ([2, 3]) *Computing a MNE of a bimatrix game is a $PPAD$-complete problem.*

Theorem 7.1 is one of the greatest hits of algorithmic game theory. Its proof is far too technical to describe here; for overviews in order of increasing levels of detail, see [13, §4.2], [12, pp. 41–45], and [4].

8 Discussion and Open Questions

One interpretation of Theorem 7.1, which is not without controversy, is that the seeming intractability of the Nash equilibrium concept renders it unsuitable for general-purpose behavioral prediction. If no polynomial-time algorithm can compute a MNE of a game, then we don't expect a bunch of strategic players to find one quickly, either. More generally, in classes of games of interest, polynomial-time tractability of computing an equilibrium can be used as a necessary condition for its predictive plausibility.

Intractability is not necessarily first on the list of the Nash equilibrium's drawbacks. For example, its non-uniqueness already limits its predictive power in many settings. But the novel computational intractability critique in Theorem 7.1 is one that theoretical computer science is particularly well suited to contribute.

If we don't analyze the Nash equilibria of a game, then what should we analyze? Theorem 7.1 suggests shining a brighter spotlight on computationally tractable classes of games and equilibrium concepts. For example, our convergence results for no-regret dynamics motivate identifying properties that hold for all correlated or coarse correlated equilibria.

One natural equilibrium concept whose computational complexity remains poorly understood is ϵ-approximate MNE of bimatrix games. After translating and scaling all player payoffs so that they lie in $[0, 1]$, such an equilibrium is, by definition, a pair of mixed strategies so that neither player can increase its payoff by more than ϵ via a unilateral deviation. It is known that an ϵ-approximate MNE of a bimatrix game can be computed in polynomial time when $\epsilon \approx \frac{1}{3}$ [14] and in quasi-polynomial time when ϵ is an arbitrarily small constant (see [7] and the Problems).[10] An interesting open question is whether or not an ϵ-approximate MNE of a bimatrix game can be computed in polynomial time for arbitrarily small constants ϵ.

Another fundamental question that is poorly understood is: how hard are $PPAD$ problems, anyways? In the absence of an unconditional proof about whether or not $PPAD$ problems are polynomial-time solvable, it is important to relate the assumption that $PPAD \not\subseteq P$ to other complexity assumptions stronger than $P \neq NP$ — for example, to cryptographic assumptions like the existence of one-way functions.

[10] This quasi-polynomial-time algorithm enumerates approximations of all MNE, and in particular can identify an approximate MNE with total payoff close to that of the best MNE. Intriguingly, this harder optimization problem can be connected to the infamous planted clique problem [1, 6, 9].

References

[1] P. Austrin, M. Braverman, and E. Chlamtac. Inapproximability of NP-complete variants of Nash equilibrium. *Theory of Computing*, 9:117–142, 2013.

[2] X. Chen, X. Deng, and S.-H. Teng. Settling the complexity of two-player Nash equilibria. *Journal of the ACM*, 56(3), 2009.

[3] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[4] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

[5] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[6] E. Hazan and R. Krauthgamer. How hard is it to approximate the best Nash equilibrium? *SIAM Journal on Computing*, 40(1):79–91, 2011.

[7] R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM Conference on Electronic Commerce (EC)*, pages 36–41, 2003.

[8] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[9] L. Minder and D. Vilenchik. Small clique detection and approximate Nash equilibria. In *APPROX-RANDOM*, pages 673–685, 2009.

[10] J. F. Nash. Equilibrium points in N-person games. *Proceedings of the National Academy of Science*, 36(1):48–49, 1950.

[11] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[12] C. H. Papadimitriou. The complexity of finding Nash equilibria. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 2, pages 29–51. Cambridge University Press, 2007.

[13] T. Roughgarden. Computing equilibria: A computational complexity perspective. *Economic Theory*, 42(1):193–236, 2010.

[14] H. Tsaknakis and P. G. Spirakis. An optimization approach for approximate Nash equilibria. *Internet Mathematics*, 5(4):365–382, 2008.

[15] B. von Stengel. Equilibrium computation for two-player games in strategic and extensive form. In N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 3, pages 53–78. Cambridge University Press, 2007.