CS 684 Algorithmic Game Theory	**Scribe: Eva Tardos**
Instructor: Eva Tardos	**Monday, Jan 26- Friday, Febr 6, 2004**

1 Simple Games and some definitions

We start with discussing a few simple games. For small and simple games, the easiest way to describe a game is by a payoff matrix. Consider for example, the game *Battle of Sexes*. The two players (a boy and a girl) want to go to either a softball or a baseball game. One player prefers softball the other prefers baseball, but they both prefer to be with each other, more than they care about which game to attend. This can be expressed by a simple game matrix: both players have two possible options (referred to as strategies), namely they can go to the softball or to the baseball game. In the following matrix we give the *payoff* for each player depending on what they both choose to do, where the first and second numbers represent the payoff for the two players respectively.

	B	S
B	$2,1$	$0,0$
S	$0,0$	$1,2$

A *pure strategy* for a player is a choice of one of the options he/she can do. We say that strategies s_i for each player i form a *Nash equilibrium* if no player has the unilateral incentive to change his/her strategy. More formally, if there is no player i and strategy $\bar{s}_i$, such that player i switching to strategy $\bar{s}_i$ while all other players j staying with their strategies s_j would improve i's payoff. The idea is that a Nash equilibrium is a stable state of the game. If the players agree to play this equilibrium, then neither of them has an incentive to deviate from this agreement. The above game of the battle of sexes has two equilibria, either both players should attend the baseball game, or they should both attend the softball game.

Not all games have such Nash equilibria using pure strategies, to this end consider the matching pennies, where the two players put out a penny "head" or "tail". One wants the pennies to match, the others for them not to match. The payoff matrix is

	H	T
H	$-1,1$	$1,-1$
T	$1,-1$	$-1,1$

Later in the course, we will talk about randomized equilibria, but for now, we just want to point out that this game has no deterministic (pure strategy) Nash equilibrium. Another interesting game is the coordination game, with the following payoffs.

	C	D
C	$2,2$	$0,0$
D	$0,0$	$1,1$

This game again has two Nash equilibria: either both players agree to coordinate (play strategy C), or they both defect (play strategy D). Both playing strategy D seems like a bad idea, as they both get lower payoff, but, it is also a Nash equilibrium, as no player has the incentive unilaterally change his strategy.

Finally, maybe the most famous of these small two person games is the *prisoner's dilemma*:

	C	D
C	$3,3$	$0,4$
D	$4,0$	$1,1$

In this game the usual story is about two criminals cooperating with each other, or defecting, and telling the truth to the police. The only Nash equilibrium in this game is for both players to defect, as no matter what the other player does, each player is a little better off defecting. However, in a way they would be both better off cooperating.

To make the issues that came up in these games precise, we will define the social value of the solution to be the sum of all players payoffs. So in the prisoner's dilemma, the solution with best social value is to have both players cooperate (for a value of 6), while the only Nash equilibrium has social value of 2. In the Battle of Sexes game both Nash equilibria has social value of 3 (which is also best possible outcome), while in the coordination game one Nash had a social value of 4, while the other one had a value of 2. Note that in place of social value we can (and will) also consider other objective functions. One natural alternate function that is very common in CS, is fairness, usually formalized with the max-min objective. So the goal is to find a solution where all players have a high payoff. So the min-max value of both Nash equilibria in the battle of sexes game is 1, as one of the players is only getting a payoff of 1.

In this course we will consider games motivated by some computer science context. Examples will include routing on the Internet, load balancing of servers, cost-sharing for network design among others. For the first part of the course, we will focus on Nash equilibria, our main questions will be the following.

- Does the game have a pure Nash equilibrium? is it unique?
- How does the (social) value of a Nash equilibrium compare to the best possible outcome?
- Can we show that the value of the best Nash equilibrium is good compared to the best possible outcome.

2 Load Balancing Game

The first game we will consider is the following load balancing game. There are m servers, and n jobs. Think of the servers as serving Web pages (i.e., this group of servers is what may be used to support cnn.com), and the jobs as requests for some pages. A job j will have a load or job size p_j, and a subset S_j of servers that it is allowed to go on (say where the appropriate information is available, or servers that is physically near enough to the request). We will think of each job as a selfish player. A solution is an assignment to jobs to servers, and the load of a server i is the sum of the loads of the set of jobs assigned to i. More formally, let M be an assignment of jobs to servers, so $(j, i) \in M$ if job j is assigned to server i. Then we define the load of server i as

$$L_i = \sum_{j:(j,i)\in M} p_j.$$

The higher the load is the worse the response rate of the server will be. If all servers are identical, then we has that the assignment is at *Nash equilibrium*, if for all jobs j, if j is assigned to server i and $k \in S_j$ is an other server that could serve this job, than $L_i \leq L_k + p_j$, that is the load on server i is no worse, than the load on the other server k would be if jobs j would switch to it.

Later we will consider cases when the servers are not identical. Assume each server i has a load dependent response time: let $r_i(x)$ be the response time on server i as a function of the load x on the server. We assume that the function is nonnegative, and monotone increasing in the load x. With such response time as assignment is a Nash equilibrium if for all jobs j, if j is assigned to server i and $k \in S_j$ is an other server that could serve this job, than $r_i(L_i) \leq r_k(L_k + p_j)$.

First we want to show that in this load balancing game a pure strategy Nash equilibrium always exists.

Theorem 1 *If the response time of each server i is a nonnegative, and monotone function of the load, then the game has a pure strategy Nash equilibrium.*

Proof. The idea is to start with any assignment, and let the jobs migrate as they desire. We will show that this process terminates. When no job wants to move that is a Nash by definition.

Consider an assignment M, and its set of response times, $\{r_i(L_i)\}$. If job j migrates from server i to server k, then L_i decreases while L_k increases. To show that this migration process terminates we need to find some quantity that improves. We will not need to worry about how much the improvement is: As long as there is some improvement the process cannot cycle, the same assignment cannot be reached again. This implies that after going through possibly all assignments, the process will terminate.

First note that the worst response time $\max_i r_i(L_i)$ is monotone improving as jobs migrate. Unfortunately, this quantity can be unchanged, if either the job moving was not on the server with worst response time, or if there are many servers with equal response time.

The idea is as follows. When the worst response time is unchanged, than consider the second highest response time. Now this one either decreases or is unchanged. More generally, assume that a job j moves from server i to some other server k. Assume that we order the servers so that $r_1(L_1) \geq r_2(L_2) \geq \ldots \geq r_m(L_m)$, and that server i is the last among equal response time servers. We need to notice two facts.

- In the above order $k > i$
- $r_i(L_i) > r_k(L_k + p_j)$ by the definition of a selfish move for job j.

The response time on servers ℓ with $\ell \neq i, k$ are not effected by the change. If we reorder the servers so that they will again be in decreasing order of response time, then the first $i - 1$ servers will remain the same, and the new ith server, will either be the current server i, or server $i + 1$, or server k. In either case the ordered sequence of response times decreased, as the first $i - 1$ coordinates are unchanged, and the ith coordinate decreased. ■

Note that this proof provides a finite algorithm to find a Nash equilibrium, but the algorithm can be rather slow. All we see is that it will terminate after possibly going through an exponential set of possible assignments.

Next we want to consider the quality of a Nash equilibrium. For now we will consider the worst response time as our quality measure, that is $C(M) = max_i r_i(L_i)$. First, it is not hard to prove a bound on the quality of the best Nash.

Theorem 2 *In the load balancing game, if the response time of each server i is a nonnegative, and monotone function of the load, then the game has a pure strategy Nash equilibrium with $C(M)$ minimum over all assignments M.*

Proof. Consider an assignment M' minimizing the cost $C(M')$. Now use the proof above for the existence of Nash starting from assignment M'. Selfish moves do not increase the worst response time, so the Nash assignment M obtained will satisfy $C(M) \leq C(M')$, and hence it is also optimal. ■

To prove a bound on the worst possible Nash, we will restrict our attention to the case when the response time is equal to the load itself, $r_i(x) = x$ for all i, and all jobs may be assigned to all machines.

Theorem 3 *Consider a load balancing game with $r_i(x) = x$ for all servers i, and $S_j = S$ the set of all possible servers for all jobs. In this case, all Nash equilibria M have $C(M) \leq 2min_{M'}C(M')$, where this minimum is over all possible assignments M'.*

Proof. Consider a Nash equilibria M, and the max loaded machine i. Let j be a job assigned to this machine i. The job j does not want to migrate, so that $L_k + p_j \geq L_i$ for all other servers k. Summing over all servers, we get that $\sum_k L_k \geq m(L_i - p_j)$, or rearranging terms, we have that $C(M) = L_i \leq (1/m)\sum_k L_k + p_j$.

On the other hand, for all assignments M' job j is assigned to a server, so $C(M') \geq p_j$. Also, the sum $\sum_k L_k$ is the total processing time $\sum_j p_j$ for all assignments. So the average load of a machine is always $(1/m)\sum_h p_h$, and hence $C(M') \geq \sum_h p_h$ by the max being at least the average.

Putting these two together we get that

$$C(M) = L_i \leq (1/m)\sum_k L_k + p_j = (1/m)\sum_h p_h + p_j \leq C(M') + C(M').$$

■

3 Potential Games: load balancing

The theorem we proved so far about the quality of the Nash solution are about the worst response time. Next we want to consider social welfare, that is the sum (or average) response time, as our quality measure. First note that our proofs for the quality of best and worst Nash do not extend to this objective function. For example, if we start off with the assignment that minimizes social welfare, a selfish move may make the social value worse. This happens as a selfish move improves the players objective value, but makes a machine more loaded, and which causes larger response time for a large number of other players.

We will consider this in the special case when all jobs are the same, that is $p_j = 1$ for all j. Taking the sum (or unweighted average) over the response times seems like a rather unfair objective function in the case when sum jobs are big and hence adding them to a server has major effect on all other users experience. Notice that the load of a server in this case is just the number of jobs assigned to the server. We will assume that jobs j may be only allowed to be served by a subset of servers S_j, and each server i has its own load dependent response time

We will introduce a new technique by showing that in this special case the game is a potential game. Consider the following function for an assignment M with load L_i on server i:

$$\Phi(M) = \sum_i \sum_{k=1}^{L_i} r_i(k).$$

To understand this function, let us also write the sum of response times, the social cost of our solution M in a similar form: $C(M) = \sum_i L_i r_i(L_i)$. This is true, as server i has L_i assigned jobs, and each of these jobs have a response time of $r_i(L_i)$. The potential function $\Phi(M)$ does not have a direct meaning like the social cost $C(M)$. Instead, it views jobs as arriving to each machine one-by-one, and changes the jobs arriving as kth, the response time of $r_i(k)$ with the "current" load of k, even though all of them will have response time $r_i(L_i)$. The key idea about the potential function is that it tracks the users change of payoff, or utility.

Theorem 4 *Given an assignment M, and consider the alternate assignment M' when a user j changes from server i to server k. Then the change in potential function is exactly the same as the change in player i's response time. More formally, $\Phi(M) - \Phi(M') = r_i(L_i) - r_k(L_k + 1)$.*

Proof. The change decreases the load on server i by 1, and increases the load on server k by one, so the terms in the sum changes by $r_i(L_i)$ and $r_k(L_k + 1)$ respectively by definition. ■

Note that the change of a job j from a server i to another server k effects the loads and hence the response times to all other jobs assigned to these servers. However, the potential function Φ is only sensitive to the change for the player who moved. This is the definition of a potential game. A game is a *potential game* is there is a potential function Φ such that if player i changes his strategy from s to s' with all other players unchanged, than the change in potential function is exactly the same as the change in player i's payoff.

We already know from last time that our load balancing game has a Nash equilibrium, but it is useful to state in general the following theorem.

Theorem 5 *All potential games over finite sets of strategies have Nash equilibria, namely the minimum of the potential function Φ is a Nash equilibrium.*

Proof.If a solution is not at Nash equilibrium then there is a player i who can improve his payoff by changing strategies (i.e., changing his path). By definition of a potential game, the new solution obtained by this change has improved objective function value. ■

This new proof for the existence of the Nash will be useful to show a bound on the quality of the best Nash equilibrium.

Theorem 6 ***If a potential game with cost C and potential Φ satisfies that $\Phi(M) \leq C(M) \leq \alpha\Phi(M)$ for all solutions M and some value α, then there is a Nash equilibrium M so that $C(M) \leq \alpha min_{M'} C(M')$, where the minimum is over all possible solutions.***

Before we prove our theorem, note that the condition $\Phi(M) \leq C(M)$ is true for our potential function, as we assumed that the response time is a monotone increasing function of the load. The value α depends on what the response function is. For example, if $r_i(x) = x$, then it is not hard to see that $\alpha = 2$ works, as $1 + 2 + \ldots + L \leq L^2/2$.

Proof. Let M be the solution that minimizes $\Phi(M)$. From the above theorem, we know that M is a Nash. We will prove that $C(M) \leq \alpha min_{M'} C(M')$, which proves the claim. The idea is that by the assumption C and Φ are the same within some error factor α. We get M be minimizing Φ rather than C, so we may be making an error by this factor α compared to minimizing the real function C. More formally, we can write the following chain of inequalities, where M' is any other solution.

$$C(M) \leq \alpha\Phi(M) \leq \alpha\Phi(M') \leq C(M').$$

■.

4 Routing Game

Next, we will consider a more general game, the routing game on graphs. This game is a simple model of routing on the Internet. So the game is given by a directed graph $G = (V, E)$, and k source-sink pairs $s_i, t_i \in V$ for $i = 1, ..., k$. The problem will be to find paths P_i from s_i to t_i in the graph. Given a set of paths, let $f(e)$ be the number of paths using edge e, which we will also think of as the load of edge e. There are one more sets of inputs: each edge e has a load dependent delay function $d_e(x)$ is the delay of edge e if there are x paths using this edge. So the delay on a path P is now $\sum_{e\in P} d_e(f(e))$.

We will think of the graph and the source sink pairs as given. The players are associated with the source-sink pairs, and the goal of player i is to have its paths P_i as small a delay as possible.

First note that the load balancing game with equal jobs that we have studies last time is a special case of this routing game. To see this we consider the bipartite graph, where jobs are on one side, servers on the other, and job j is connected to server i by an edge if server i can serve job j. These edges will have 0 delay. Recall that load balancing has delays (or response time) on servers. To model this as an edge delay we introduce a new note t, which will be the common destination for all players, and have $r_i(x)$ be the delay on edge (i, t). Now the routing game source sink pairs (i, t) is exactly the load balancing game.

It is a bit more awkward to state the condition what makes a set of paths P_i for $i = 1, ..., k$ for this problem a Nash equilibrium. Consider a path P_i and an alternate path Q_i connecting s_i to t_i. By changing the path from P_i to Q_i the load on the edges in $P_i \cap Q_i$ are not effected, while the load on the edges $Q_i \setminus P_i$ increases by 1. So the solution is at Nash equilibrium, if the following holds for all i and all alternate paths Q_i.

$$\sum_{e\in P_i\setminus Q_i} d_e(f(e)) \leq \sum_{e\in Q_i\setminus P_i} d_e(f(e)+1).$$

It is not hard to see that the potential function for our load balancing game extends to the routing game.

Theorem 7 ***The above routing game is a potential game, where the potential is of a solution is*** $\Phi = \sum_e \sum_{k=1}^{f(e)} d_e(k)$.

Proof. Again the proof is essentially by definition: when a player changes from path P_i to an alternate path Q_i, then for edges $e \in P_i \setminus Q_i$ the last term $d_e(f(e))$ is deleted, and for edges $e \in Q_i \setminus P_i$ a new last term of $d_e(f(e)+1)$ is added, the term corresponding to all other edges is unaffected. This is exactly the change on delay of player i. ■.

For the cost of a solution, we will again consider social welfare, which is defines as $C(f) = \sum_i \sum_{e \in P_i} d_e(f(e))$. It is useful to write the social cost in a form that is analogous to the potential function.

Lemma 8 *The social cost of a routing can be written also as* $C(f) = \sum_e f(e) d_e(f(e))$.

Proof. By definition $C(f) = \sum_i \sum_{e \in P_i} d_e(f(e))$, we get the alternate expression, by swapping the order of summation, and noticing that the term $d_e(f(e))$ is contained in $f(e)$ number of paths i. ■

Using our theorems from last time, and noticing from the above lemma that $C \leq \Phi$ for all routings, we get the following.

Theorem 9 *The routing game defined above always has a Nash, and the best Nash has cost at most* $\max \Phi / C$*, where the max is over all possible routings.*

Finally, we will consider one other question: can we find a Nash equilibrium in polynomial time in this game? First assume that all players have the same source and sink.

Theorem 10 *If all players have the same source and sink* (s,t) *then a Nash equilibrium can be found in polynomial time using a minimum cost flow computation.*

Proof.The equilibrium we will find, is the one that minimizes the potential Φ. To this end, consider a copy of the original graph G where each edge e is replaced by k parallel edges (where k is the number of players) each with capacity 1, and the kth copy of the edge costs $d_e(k)$. As the edge cost are monotone increasing, if a flow sends some x units along copies of the edges e, it will always use the first x parallel copies, as those are cheaper. So, the minimum cost set of k disjoint paths in this graph corresponds to a solution that minimizes the cost Φ of the solution. ■

Now consider the case when different players have different source-sink pairs. The same construction works to show that the solution minimizing Φ can be found by finding a minimum cost set of paths. However, finding the minimum cost set of paths with multiple source-sink pairs is NP-complete, and hence this does not lead to a polynomial time solution. it would be very interesting if one could show that finding a Nash equilibrium is also NP-complete. However, this may be hard: Nash equilibria are more analogous to local optima in local optimization, and no such hardness result are known for local optimization. Fabrikant, Papadimitriou and Talwar showed that the finding a Nash equilibrium in this game is PLS complete, where PLS is a class of problem for finding local optima in optimization problems.

5 Non-atomic Games

So far we assumed that each user of our game is routing one unit of traffic along its selected path. We had some trouble, or at least awkwardness over the effect of

additional user as he changes path (recall the terms $d_e(f(e)+1)$ last time. In many cases, traffic on a real edge is very big, and the effect of a single additional user is essentially negligible. The following non-atomic game is trying to model this. As before, it will be a routing game, with all source-sink pairs need to send a unit of traffic from the source to the sink, but we will think of the one unit of flow as being controlled by a huge number of players, each pursuing their selfish goal separately.

The input is a graph $G = (V, E)$ and source-sink pairs (s_i, t_i) as before, and a delay function $d_e(x) \geq 0$ for each edge. We will assume that the delay on an edge is a monotone increasing and continuous function of x, which will be the amount of traffic on the edge. To model traffic being sent on many paths all at once, let $\mathcal{P}_i$ denote set of paths some s_i to t_i, and we define a flow (or a solution) as follows. The flow has a value $f_P \geq 0$ for all $P \in \cup_i \mathcal{P}_i$ such that $\sum_{P\in\mathcal{P}_i} f_P = 1$ for all i. We think of f_P as the amount of flow carried directly along P from its source to its sink. And the condition $\sum_{P\in\mathcal{P}_i} f_P = 1$ ensures that the total flow carried from a source to the corresponding sink is 1. We use $f(e) = \sum_{P:e\in P} f_P$ to denote the amount of flow carried along edge e, and now the delay of a path is $d_P(f) = \sum_{e\in P} d_e(f(e))$.

We assume that each selfish user wants to have his (infinitesimally) small amount of flow to be carried on a path with as small delay as possible. So we define a solution to be at *Nash equilibrium* is for all i and all $P \in \mathcal{P}_i$ with $f_P > 0$ and all alternate paths $Q \in \mathcal{P}_i$ we have that $d_P(f) \leq d_Q(f)$. We use this as our definition, but note that it has a strong connection to the Nash equilibrium definition we used in finite games: if this condition fails than the many small users flow is carried along P would all prefer to switch to the alternate path Q. Our delay functions are continuous and each user carries a negligible amount of flow, so the delay along Q will (essentially) not increase if any one of these users switches to the path Q.

The main benefit for using non-atomic games is that we will be able to rely on convex optimization as a tool. We will be able to show that the Nash equilibrium exists, it is unique, and will be able to bound how far it is in social value compared to the best possible routing. The main tool that we'll use is the following. Suppose there is some function $c_e(x)$ for all edges e, and assume $c_e(x)$ is a convex function of x. Now define a corresponding objective function for flows as $C(f) = \sum_e c_e(f(e))$. This is now a multivariate convex function over the convex space of all flows. (Note that if f and f' are flows, then we can define a new flow as $(f_P + f'_P)/2$ for all paths P. This also defines a flow, and will have flow on edge e as $(f(e) + f'(e))/2$. This shows that the space of all flow vectors is convex.

Theorem 11 (Convex Optimization) ***A flow f minimizes the convex function $C(f)$, with costs $c_e(x)$ differentiable, if and only if for all $P \in \mathcal{P}_i$ such that $f_P > 0$ and all $Q \in \mathcal{P}_i$ we have that***

$$\sum_{e\in P} c'_e(f(e)) \leq \sum_{e\in Q} c'_e(f(e)),$$

where $c'_e(x)$ is the derivative of function $c_e(x)$ at x.

We'll review what we need from convex optimization next time, including some sense of why this theorem is true. For now we will use this, along with the fact that

the minimum always exists, to show that the non-atomic routing game we defined has a Nash equilibrium. Our proof is along the same lines as our potential game proof for last time. The natural extension of the potential Φ from last time to non-atomic games is the following.

$$\Phi(f) = \sum_e \int_0^{f(e)} d_e(x)dx.$$

Lemma 12 *If the delay $d_e(x)$ is continuous and monotone increasing, then the cost $c_e(y) = \int_0^y d_e(x)dx$ is a differentiable, and convex function.*

Proof. By basic properties of the integral, $c_e(y)$ is differentiable, and $c_e'(x) = d_e(x)$. The function is convex as its derivative is monotone increasing. ■

Theorem 13 *If the edge delay functions $d_e(x)$ are monotone and increasing, then the non-atomic game has a Nash equilibrium, the flow that minimizes $\Phi(f)$.*

Proof. By the above lemma $c_e(x)$ is a convex function, and hence we can use the Convex Programming theorem above to conclude that Φ has a minimum, and a flow f minimizes Φ if and only if $\sum_{e\in P} c_e'(f(e)) \le \sum_{e\in Q} c_e'(f(e))$ for all flow path P (that is paths P with $f_P > 0$), and all alternate path Q of the same source-sink pair. Recall that $c_e'(x) = d_e(x)$ again by the lemma, so a flow f is optimal for Φ if and only if $\sum_{e\in P} d_e(f(e)) \le \sum_{e\in Q} d_e(f(e))$. This is exactly the condition that defines Nash equilibria. ■

6 What we need to know about convex programming

A fairly general definition of convex optimization is as follows. Let $K \subset R^N$ be a closed, convex set. Recall that convex means that for all $x, y \in K$ in $(x + y)/2$ is also in K. Now convex optimization is concerned by minimizing a convex function $\Phi(x)$ over $x \in K$. Here we will not be concerned with general convex function, rather only separable functions, functions defined as $\Phi(x) = \sum_i c_i(x_i)$. If each function c_i is a convex function of its one variable x_i then the multivariate function Φ is convex. Generally, of course not all convex function can be written as a separable sum of functions of single variables, but the functions we will care about will have this form.

Last time we considered the function $\Phi(f)$ over the space of all flows f. Recall, as we already noted there that the space of all flows is convex. Note now that the function Φ is convex and separable (and it has no terms associated to the variables f_P). It is interesting to consider the dimension of the space defining the convex optimization whose optima are Nash equilibria. So we need the flow variables f_P to define the solution. Given the flow variables, they uniquely determine the edge flows $f(e)$, but we added them as separate coordinates, as we want our objective function Φ to be separable.

Now we want to state the form of the convex optimization theorem that applies to minimizing separable convex functions over a convex set K.

Theorem 14 (Convex Optimization) ***Consider a convex separable function $\Phi(x) = sum_i c_i(x_i)$ where $c_i(x)$ is differentiable for all i over a convex set K. A point x in K minimizes $\Phi(x)$ if and only if, for all directions y such that $x + \epsilon y \in K$ for a small enough $\epsilon > 0$, we have that $\sum_i c_i'(y_i)y_i \geq 0$. If K is bounded, or if the value $\Phi(x)$ is bounded from below over the set K (e.g., non-negative), then the minimum exists. Further if x^* and x' both minimize Φ, then each of the coordinates $c_i(x)$ is linear in the $[x_i^*, x_i']$ interval.***

The theorem states a few different things: (1) sufficient and necessary conditions on when does a solution x minimize Φ, (2) states that the minimum exists, and (3) states that if there are multiple minima then each function c_i is linear in the corresponding range. Here linear means that for each i there are coefficients a_i and b_i so that for all $x \in [x_i^*, x_i']$ $c_i(x) = a_i x + b_i$.

Proof of one direction. We will not prove this theorem. One direction of the if and only if statement is easy, and it helps us understand the condition, so we will sketch the proof of this easy direction of the first statement.

If a function of differentiable than $c(x + \epsilon) \approx c(x) + \epsilon c'(x)$ is true for all small enough values of ϵ. Now we will use this to evaluate $\Phi(x + \epsilon y)$. If ϵ is small enough, than ϵy_i is also small for all coordinates i, so we have the following.

$$\Phi(x+\epsilon y) = \sum_i c_i(x_i+\epsilon y_i) \approx \sum_i (c_i(x_i)+\epsilon y_i c_i'(x_i)) = \sum_i c_i(x_i)+\epsilon \sum_i y_i c_i'(x_i) = \Phi(x)+\epsilon \sum_i y_i c_i'(x_i).$$

So if $\sum_i y_i c_i'(x_i) < 0$ then we can choose a small enough ϵ so that the new value $\Phi(x + \epsilon y)$ is smaller, contradicting to the assumption that x minimizes Φ. ■

One final important point to make here is that convex optimization problems can be solved in polynomial time. To be precise, finding the exact optimum of a general convex function may be hard, as for example, the optimum may be obtained at an irrational point x. What we'll use instead is the following weaker statement. Again, we will not prove this theorem

Theorem 15 ***For any function $\Phi(x) = \sum_i c_i(x_i)$ such that c_i is convex for all i, and any closed convex set K, an approximate minimum of the function $\Phi(x)$ can be found over $x \in K$ in time polynomial in the dimension and the size of the input numbers, assuming we have a polynomial time separation algorithm for K.***

Here a separation algorithm is an algorithm that can decide of a given point x is in K, and if it is not, give a direction a such that $ax > max_{y \in K} ay$.

We will use the last two theorems for flows, and so it make sense to derive its simplest form if the convex set K in question is the set of all flows. We'll start our discussion with the second statement. A separation algorithm for the convex set of flows is easy, flows are described by the simple equations $\sum_{P \in \mathcal{P}_i} f_P = 1$ and the non-negativity of the variables. However, the statement of the theorem is not directly useful for flows, as flows the dimension of the problem we defined is the set of all possible paths, which can be exponential in the natural size of the problem (size of graph plus the size of numbers). However, the functions we will consider, will always be of the form $\sum_e c_e(f(e))$, that is will not depend on the path variables, only the edge flows. Using convex programming techniques one can find close to optimal solution for this class of problems in polynomial time.

Theorem 16 *For any function $\Phi(f) = \sum_e c_e(f(e))$ for a flow problem, an approximate minimum of the function $\Phi(f)$ can be found in time polynomial in the size of the graph and the size of the input numbers.*

The first theorem (the optimality condition) considers a flow $f = x$, and direction y, so that $x + \epsilon y$ is feasible. One possible "elementary" change direction y that we take flow off from a path P that carries flow (has $f_P > 0$) and place the same amount of flow on an alternate path connecting the same set of terminals. It is not hard to see that all possible change from one flow f to some other flow f' can be written as a combination of such elementary changes. This yields Theorem 11 we stated last time as convex optimization theorem.

We will use this theorem for two different functions. Last time we considered the potential function Φ where the derivative of the component of edge e was $d_e(x)$. Another important function that we will consider is the social welfare function $C(f) = \sum_P f_P d_P(f)$. In this form the function does not appear to be separable, as the value $f_P(f)$ depends on flow values along many paths. Recall that this function can also be written as $C(f) = \sum_e f(e) d_e(f(e))$, and this form is clearly separable.

Theorem 17 *If the function $x d_e(x)$ is convex, and $d_e(x)$ is differentiable for all edges, then the flow f is optimal if and only of for all paths $P \in \mathcal{P}_i$ such that $f_P > 0$, and all alternate paths $Q \in \mathcal{P}_i$, we have the following inequality:*

$$\sum_{e \in P} (d_e(f(e) + f(e) d'_e(f(e))) \leq \sum_{e \in Q} (d_e(f(e) + f(e) d'_e(f(e))).$$

Proof. To derive the theorem from convex optimization, all we have to notice is that the derivative of $c_e(x) = x d_e(x)$ is exactly $c'_e(x) = d_e(x) + x d'_e(x)$ by the chain rule. ■

Note the similarity of the Nash condition, and the optimality condition for social welfare. The difference is the additional term of $f(e) d'_e(f(e)$. The Nash flow simply (and selfishly) chooses the path with smallest delay, while the socially optimal flow evaluates paths in a socially aware way: if additional flow is added to edge e then the $f(e)$ units of flow currently on the edge will see their delay increase at a rate of $d'_e(f(e))$. The "socially aware flow" evaluates edge by adding a second term to the delay that corresponds to the effect of his change on others.

CS 684: Algorithmic Game Theory	Scribe: Jonathan Winter
Instructor: Éva Tardos	Monday February 9, 2004

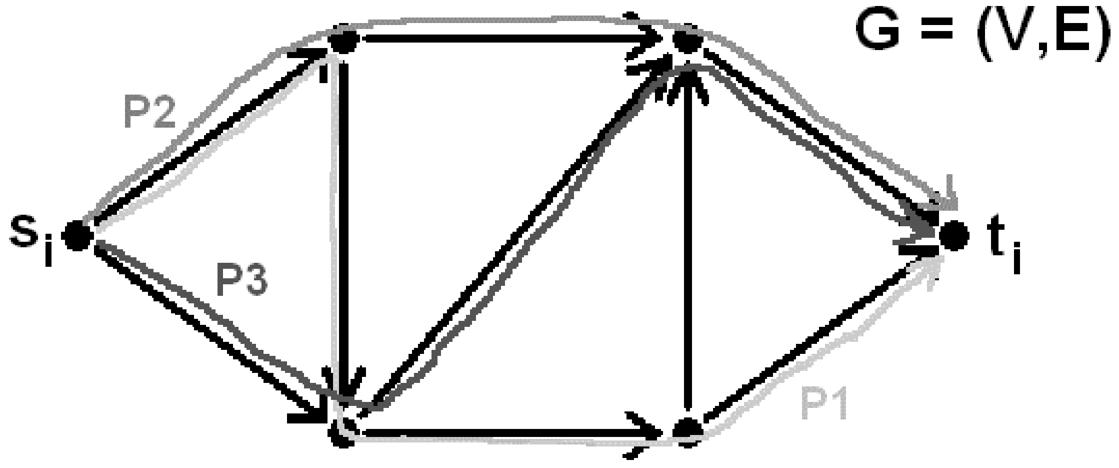
Worst Case Nash / Optimal Ratio

What we learned from convex optimization:

APPLICATION:

- Let G = (V, E), with a continuous and monotonic delay function, $d_e(x) \geq 0$, for each edge $e \in E$.
- Let s_i, t_i be source, sink pairs for i = 1, 2, …, k and $\boldsymbol{P}_i$ = {set of all paths from s_i to t_i}.
- Define flow $f_P \geq 0$ for $P \in \bigcup_i \boldsymbol{P}_i$ with the property that $\sum_{P \in P_i} f_P = 1$.

Now, the flow on edge e is $f(e) = \sum_{P,\, e \in P} f_P$. Also, delay on P is $d_P(f) = \sum_{e \in P} d_e(f(e))$.



The flow at Nash Equilibrium requires that $\forall P \in \boldsymbol{P}_i$, if $f_P > 0$ and $Q \in \boldsymbol{P}_i$, then

$$d_P(f) \leq d_Q(f). \qquad \text{(A)}$$

(The logic behind this is that no user on a path P wants to switch to any other path.)

THEOREM 7.1: Suppose the goal is to minimize $\sum_{e \in E} c_e(f(e))$, where c_e is convex and differentiable. (Note: the summation is separable.) Then, the flow f is optimal if and only if $\forall P \in \boldsymbol{P}_i$, if $f_P > 0$ and $Q \in \boldsymbol{P}_i$, then

$$\sum_{e \in P} c_e'(f(e)) \leq \sum_{e \in Q} c_e'(f(e)). \qquad \text{(B)}$$

(Note: here c_e' is the derivative of c_e.)

COROLLARY 7.1a: Nash Equilibrium is the optimal flow and of course optimizes $\text{Ø}(f)$, where $\text{Ø}(f) = \sum_{e \in E} \int_0^{f(e)} d_e(x)\, dx$. This follows by substituting

$_0\int^{f(e)} d_e(x)\,dx$ for $c_e(f(e))$ in equation (B) and noting that the derivative of the integral, $d/dx\,(\,_0\int^{f(e)} d_e(x)\,dx\,) = d_e(f(e))$. The resulting equation is

$$\sum_{e\in P} d_e(f(e)) \le \sum_{e\in Q} d_e(f(e)),$$

which is equivalent to $d_P(f) \le d_Q(f)$. Hence, by (A) our Nash Equilibrium flow satisfies the preconditions for Theorem 7.1.

COROLLARY 7.1b: The approximate Nash Equilibrium flow can be found in polynomial time.

Let us consider a new objective function, $\sum_{P\in \boldsymbol{P}_i} f_P \cdot d_P(f_P) = \sum_{e\in E} f(e) \cdot d_e(f(e))$. Assume $x \cdot d_e(x)$ is convex for all edges (this is usually true for most $d_e(x)$ functions).

COROLLARY 7.1c: If $x \cdot d_e(x)$ is convex for all $e \in E$, then the optimal flow, f, is obtained if and only if $\forall\, P \in \boldsymbol{P}_i$, if $f_P > 0$ and $Q \in \boldsymbol{P}_i$, then

$$\sum_{e\in P} (\, d_e(f(e)) + f(e) \cdot d_e'(f(e))\,) \le \sum_{e\in Q} (\, d_e(f(e)) + f(e) \cdot d_e'(f(e))\,) \qquad \text{(C)}$$

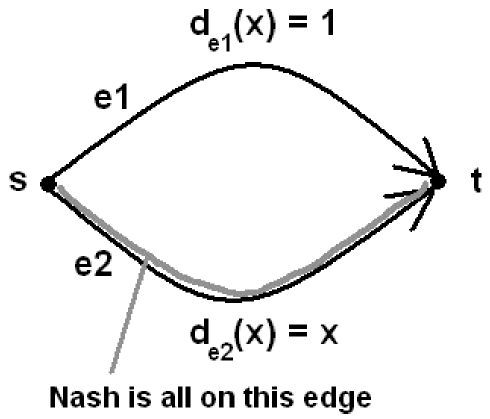
COROLLARY 7.1d: The approximate optimal flow (in an average happiness sense) can be computed if $x \cdot d_e(x)$ is convex.

COROLLARY 7.1e: For a new delay function $\boldsymbol{d}_e^*(x) = d_e(x) + x \cdot d_e'(x)$, the Nash Equilibrium flow is actually the optimal flow (in an average happiness sense) for the original routing problem. Therefore, a network administrator's strategy to achieve optimal flow could be to charge $x \cdot d_e'(x)$ as a tax/fee for using the network.

** Charging money can make people behave Nashfully. **

GOAL: Compare Nash flow with Optimal flow:

Example 1:



Nash: All flow is on lower edge with delay $d_{e2}(1) = 1$.

Optimal:

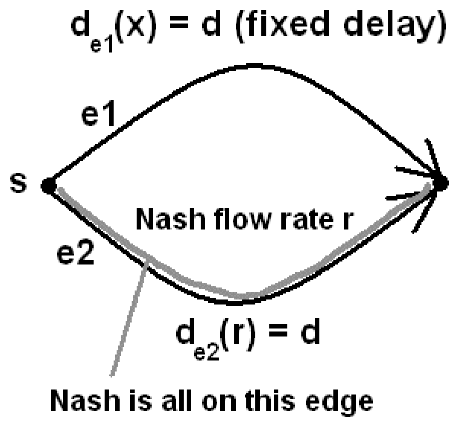
Upper edge: $\boldsymbol{d}_{e1}^*(x) = d_{e1}(x) + x \cdot d_{e1}'(x) = 1 + x \cdot 0 = 1$

Lower edge: $\boldsymbol{d}_{e2}^*(x) = d_{e2}(x) + x \cdot d_{e2}'(x) = x + x \cdot 1 = 2x$

Optimal occurs when delays are equal

($\boldsymbol{d}_{e1}*(x) = \boldsymbol{d}_{e2}*(x)$), so the flow will be split ½ on the top edge and ½ on the bottom edge.

Example 2:



Nash: All flow is on lower edge with delay $d_{e2}(r) = d$, where r is the Nash flow rate.

Optimal:

Upper edge: $\boldsymbol{d}_{e1}*(x) = d_{e1}(x) + x \cdot d_{e1}'(x)$
$= d_{e2}(r) + x \cdot d/dx\,(d_{e2}(r))$
$= d + x \cdot 0$
$= d$

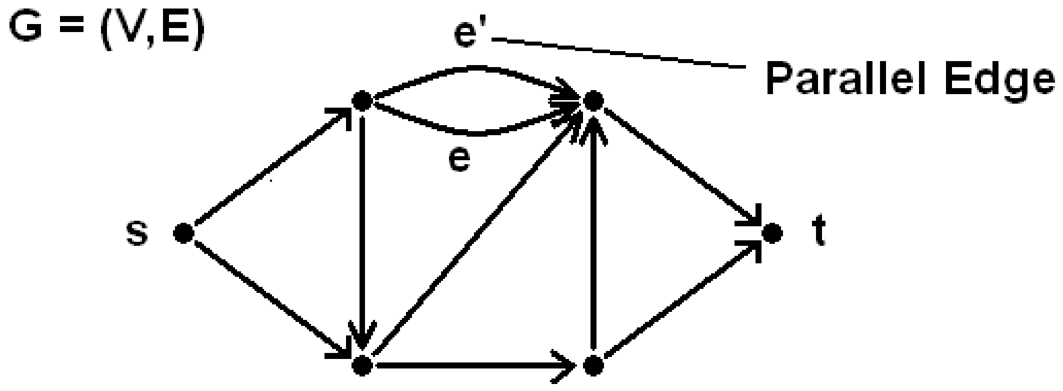
Lower edge: $\boldsymbol{d}_{e2}*(x) = d_{e2}(x) + x \cdot d_{e2}'(x)$

If r* is the flow on e2 in the optimal case, r – r* will be the flow on e1. Then, r* can be computed by solving:

$$d = \boldsymbol{d}_{e2}*(r^*) = d_{e2}(r^*) + r^* \cdot d_{e2}'(r^*).$$

THEOREM 7.2 (Roughgarden): The worst case of Nash / Optimal ratio for any class of delays, $x \cdot d_e(x)$ (convex and differentiable), is on a 2-edge, 2 node graph with one edge having a constant delay.

PROOF: Consider the graph G = (V, E) as shown.



Let f^N be the Nash flow on G. Consider G' = (V, E') created from G by adding a parallel copy to every edge $e \in E$ called e'. Let e' have fixed delay $d_{e'}(x) = d_e(f^N(e))$.

Facts:

1. f^N is still a Nash flow for G'.
2. The Optimal flow for G' may have improved over the Optimal flow for the original graph G.
3. We claim that the Optimal flow on G' is obtained from the Nash by dividing the flow between e and e' optimally as shown in Example 2.

Proof of 3: Assume $\boldsymbol{f^*}$ is the flow constructed in Claim 3 by dividing the flow $f^N(e)$ between the two parallel copies. Let $d_{e'}(x)$ to denote the (constant) delay of e', the new parallel copy of edge e. We want to claim that $\boldsymbol{f^*}$ is the optimal flow. Define the new delay function as $\boldsymbol{d}_e^*(x) = d_e(x) + x \cdot d_e'(x)$. By definition of how we divide the flow between the two copies of an edge, e and e', we have the following:

$$\boldsymbol{d}_e^*(\boldsymbol{f^*}(e)) = \boldsymbol{d}_{e'}^*(\boldsymbol{f^*}(e)) = d_e(f^N(e))$$

Therefore, $\boldsymbol{f^*}$ is the Nash flow subject to the delay function $\boldsymbol{d}_e^*$ (all flow on the shortest s_i - t_i paths). This implies that $\boldsymbol{f^*}$ is the Optimal flow for G'.

Continuing with the proof of Theorem 7.2:

$$\frac{\text{Nash}}{\text{Opt}} \ \text{(on G)} \quad \leq \quad \frac{\text{cost of } f^N}{\text{cost of } \boldsymbol{f^*}} \quad = \quad \frac{\sum_{e \in E} f^N(e) \cdot d_e(f^N(e))}{\sum_{e \in E'} \boldsymbol{f^*}(e) \cdot d_e(\boldsymbol{f^*}(e))}$$

$$\leq \max_e \frac{f^N(e) \cdot d_e(f^N(e))}{\boldsymbol{f^*}(e) \cdot d_e(\boldsymbol{f^*}(e)) + \boldsymbol{f^*}(e') \cdot d_{e'}(\boldsymbol{f^*}(e'))}$$

Notes: The first inequality follows from applying facts 1, 2, and 3 to G'.
The final inequality follows from the math theorem: $\frac{a + b}{a' + b'} \leq \max\left(\frac{a}{a'}, \frac{b}{b'}\right)$.

CS 684 Algorithmic Game Theory	**Scribe: Zoya Svitkina**
Instructor: Eva Tardos	**February 11, 2004**

1 More on nonatomic flows

We will use the following notation as usual:

G is a graph with monotone and continuous delay function $d_e(x)$ on edges. For each $i = 1...k$, there is one unit of flow from s_i to t_i. $\mathcal{P}$ is the set of $s_i - t_i$ paths, and $f_P \geq 0$ for $P \in \mathcal{P}$ is flow on P. $f(e) = \sum_{P:e\in P} f_P$ is the flow (or congestion) on edge e. The delay on a path is $d_P(f) = \sum_{e\in P} d_e(f(e))$.

We know that a flow f is at Nash equilibrium if and only if it minimizes the following objective function:

$$\Phi(f) = \sum_{e\in E} \int_0^{f(e)} d_e(x)dx$$

We would like to say something to the effect that Nash equilibrium is unique, but of course this is not true: in a two-node graph with two parallel edges of constant delay 1, any flow is at Nash equilibrium. But we do have the following.

Theorem 1 *If the delay is monotone non-decreasing and continuous, then, for any edge e, $d_e(f(e))$ is the same for any flow that is at Nash equilibrium.*

Proof. We showed before that Φ is convex. From the theory of convex optimization, we know the following two facts:

1. The set of optimal flows is a convex set; i.e., if f and f' are optimal, then so is $\alpha f + (1-\alpha)f'$, $\alpha \in [0,1]$.
2. Φ on this convex set is a linear function; i.e., it is of the form $\sum_e (a_e f(e) + b_e)$ for some a and b.

This means that $\int_0^{f(e)} d_e(x)dx$ is linear on the interval between $f(e)$ and $f'(e)$. Then its slope, or derivative, on this interval is constant and is equal to $d_e(f(e)) = d_e(f'(e))$. We conclude that the delay function for edge e is constant for flow values in $(f(e), f'(e))$. ■

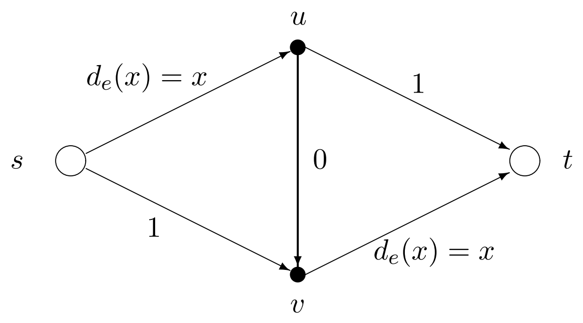
Fact. If f is a Nash flow, then for all i, $P, Q, R \in \mathcal{P}_\rangle$ with $f_P > 0$ and $f_Q > 0$,

$$d_P(f) = d_Q(f) \leq d_R(f)$$

As a result, the notion of "travel time from s_i to t_i" is well-defined, in the sense that it is the same for all paths and in all Nash equilibria.

2 Envy-free flows and Braess paradox

We can define "envy-free" flows as ones in which all paths with positive flow between the same terminals s_i and t_i have the same delay. Then the problem is to make the delay low, subject to the condition that flow be envy-free. At first sight this may seem very similar to Nash equilibrium, but that is not actually the case, as demonstrated in the following example.



If we consider the possible paths from s to v (see figure), it is clear that the selfish players will choose to take the $s - u - v$ path, as its delay is never more than 1. Similarly, to go from u to t, they would choose the path $u - v - t$. As a result, the Nash flow on this graph is for all flow to take the $s - u - v - t$ path, with the delay of 2. The optimal flow on this graph, which happens to be envy-free, is to route half a unit of flow on $s - u - t$, and half on $s - v - t$, for a delay of 1.5.

In a two-node two-edge example that we saw earlier, with delays 1 and x, the optimal flow was also split in half and had a delay of 1.5, but it was not envy-free.

2.1 Opt/Nash ratio

Our objective function here is

$$Opt = C(f) = \sum_P f_P d_P(f)$$

We had two bounds on the Nash to Opt ratio:

1. If for all f, $\Phi(f) \le C(f) \le \alpha\Phi(f)$, then $\frac{Nash}{Opt} \le \alpha$.
2. $max(\frac{Nash}{Opt}) = (max\ \frac{Nash}{Opt}$ on 2-edge, 2-node graph).

Looking at the special case when all delays are linear, i.e., for $a_e \ge 0$ and $b_e \ge 0$, $d_e(x) = a_e x + b_e$, we get the following corollaries corresponding to the two bounds.

1.

$$\begin{aligned} C(f) = &\ \textstyle\sum_e d_e(f(e))f(e) &= \sum_e (a_e f(e) + b_e) f(e) \\ \Phi(f) = &\ \textstyle\sum_e \int_0^{f(e)} d_e(x)dx &= \sum_e (\frac{1}{2} a_e f(e)^2 + b_e f(e)) \end{aligned}$$

As we can see, if d is linear, then $max\ \frac{C(f)}{\Phi(f)} \le 2$.

2. Fact: the worst-case ratio for a two-edge graph and linear delay is 4/3. We will not prove this here, but a useful idea is, instead of thinking of delay as being $ax + b$, to think of it as being *either* a *or* b. The corresponding transformation to the graph is to replace each edge of delay $ax + b$ with two consecutive edges of delay ax and b.

CS 684 Algorithmic Game Theory	Scribe: David Martin
Instructor: Eva Tardos	Friday, Feb 13, 2004

Non-atomic Routing Games

Nash Cost vs. Optimal Cost

Last time, we started analyzing worst-case Nash/Opt cost ratios for various delay functions for non-atomic routing games.

Today, we will look at two types of delay functions that can lead to arbitrarily high the Nash/Opt cost ratios.

However, we will then show that, for a modified the optimality condition, the Nash/Opt cost ratio is indeed bounded regardless of the delay function.

Polynomial Delay Functions (x^n)

Setup

Consider graph $G = (V, E)$, where $V = \{s_1, t_1\}$ and $E = \{e_1, e_2\}$ and e_1 and e_2 are distinct edges from s_1 to t_1.

The delay functions are:

$$\begin{aligned} d_{e_1}(x) &= 1 \\ d_{e_2}(x) &= x^n \end{aligned}$$

The cost function C is:

$$\begin{aligned} C(f) &= (1-x).1 + x.x^n \\ &= 1 - x + x^{n+1} \end{aligned} \quad \text{, where } x = f(e_2)$$

Nash Cost

Nash flow sends all flow on edge e_2, and hence Nash delay is 1.

Optimal Cost

Let x be the flow on edge e_2 for the optimal flow.

$$\begin{aligned} 0 &= C'(f^*) \\ &= ((1-x).1 + x.x^n)' \\ &= (n+1).x^n - 1 \end{aligned}$$

Thus $x = (n+1)^{-1/n}$.

Thus optimal cost is:

$$\begin{aligned} C(f^*) &= 1 - x + x^{n+1} \\ &= 1 - x.(1 - x^n) \\ &= 1 - (n+1)^{-1/n}.n/(n+1) \\ &= 1 - n/(n+1)^{1+1/n} \end{aligned}$$

Nash/Opt Ratio

Notice that as n increases, $1 - n/(n+1)^{1+1/n}$ goes to 0.

Hence the Nash/Opt cost ratio increases unboundedly as n increases.

Queuing Delay Functions **($1/(u-x)$, where u is the capacity)**

Setup

Consider graph $G=(V,E)$, where $V=\{s_1,t_1\}$ and $E=\{e_1,e_2\}$ and e_1 and e_2 are distinct edges from s_1 to t_1.

The delay functions are:

$$\begin{array}{rcl} d_{e_1}(x) & = & 1/\varepsilon \\ d_{e_2}(x) & = & 1/(1+\varepsilon-x) \end{array}, \text{where } \varepsilon > 0$$

Remark. The ε prevents Nash equilibrium from requiring an infinite amount of flow.

The cost function C is:

$$\begin{array}{rcl} C(f) & = & (1-x).1/\varepsilon + x.1/(1+\varepsilon-x) \\ & = & (1-x)/\varepsilon + x/(1+\varepsilon-x) \end{array}, \text{where } x = f(e_2)$$

Nash Cost

Nash flow sends all flow on edge e_2, and hence Nash delay is $1/\varepsilon$.

Optimal Cost

Let x be the flow on edge e_2 for the optimal flow.

$$\begin{array}{rcl} 0 & = & C'(f^*) \\ & = & ((1-x)/\varepsilon + x/(1+\varepsilon-x))' \\ & = & -1/\varepsilon + (1+\varepsilon)/(1+\varepsilon-x)^2 \end{array}$$

Thus $x = 1+\varepsilon-(\varepsilon.(1+\varepsilon))^{1/2}$.

Thus optimal cost is:

$$\begin{array}{rcl} C(f^*) & = & (1-x)/\varepsilon + x/(1+\varepsilon-x) \\ & = & (\varepsilon-(\varepsilon.(1+\varepsilon))^{1/2})/\varepsilon + (1+\varepsilon-(\varepsilon.(1+\varepsilon))^{1/2})/(\varepsilon.(1+\varepsilon))^{1/2} \\ & = & 2.((1+1/\varepsilon)^{1/2}-1) \end{array}$$

Nash/Opt Ratio

The Nash/Opt ratio is $(1/\varepsilon)/[2.((1+1/\varepsilon)^{1/2}-1)] = 1/[2.((\varepsilon^2+\varepsilon)^{1/2}-\varepsilon)]$.

Hence the Nash/Opt cost ratio increases unboundedly as ε goes to 0.

A Modified Optimality Condition

We have thus seen two cases (Polynomial Delay & Queuing Delay) where the worst Nash/Opt cost ratio can be arbitrarily high.

We will now consider a type of optimality for which the Nash/Opt cost ratio is always bounded by 1.

Theorem (Roughgarden-Tardos). *Let f^N be a Nash flow. Let f^* be an optimal flow for which each user sends 2 units of flow instead of 1. Then:*

$$C(f^N) \leq C(f^*)$$

Proof.

We define an artificial delay function $\widehat{d_e}$ as follows:

$$\widehat{d_e}(x) = \begin{cases} d_e(f^N(e)) & \text{if } x \leq f^N(e) \\ d_e(x) & \text{otherwise} \end{cases}$$

Notice that $\widehat{d_e}$ is continuous, non-negative, and increasing (since d_e was too).

Let $\widehat{C}$ be the cost function associated with this new delay function. That is,

$$\widehat{C}(f) = \sum_{i=1}^{k} \widehat{d_e}(f(e)).f(e)$$

We will now prove two subclaims that will help us to bound $\widehat{C}(f^*)$ from above and below in terms of $C(f^*)$ and $C(f^N)$.

Subclaim 1. *If f is a flow that sends 2 units from s_i to t_i (for each i), then:*

$$\widehat{C}(f) \geq 2.C(f^N)$$

Proof.

Recall that, since f^N is a Nash flow, for any $i = 1, \cdots, k$, the d_e-delay over any s_i-t_i path is the same (say D_i). So we have:

$$\begin{array}{rcll} C(f^N) & = & \sum_{e \in E} f^N(e).d_e(f^N(e)) & \\ & = & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_p^N.d_P(f^N) & \text{(we proved this in a previous lecture)} \\ & = & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_p^N.D_i & \text{(since } f^N \text{ is a Nash flow)} \\ & = & \sum_{i=1}^{k} D_i. \sum_{P \in \mathcal{P}_i} f_p^N & \\ & = & \sum_{i=1}^{k} D_i.1 & \text{(since } f^N \text{ sends 1 unit of flow from } s_i \text{ to } t_i) \\ & = & \sum_{i=1}^{k} D_i & \end{array}$$

Consider any flow f that sends 2 units from s_i to t_i for each $i = 1, \cdots, k$.

Now, for any edge e, by definition, $\widehat{d_e}$ is bounded from below by $d_e(f^N(e)))$.

Thus, for any s_i-t_i path P, $\widehat{d_P}$ is bounded from below by $d_P(f^N) = D_i$. So we have:

$$\begin{array}{rcll} \widehat{C}(f) & = & \sum_{e \in E} f(e).\widehat{d_e}(f(e)) & \\ & = & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_p.\widehat{d_P}(f) & \text{(we proved this in a previous lecture)} \\ & \geq & \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} f_p.D_i & \text{(since } \widehat{d_P} \text{ is bounded below by } d_P(f^N) = D_i) \\ & = & \sum_{i=1}^{k} D_i. \sum_{P \in \mathcal{P}_i} f_p & \\ & = & \sum_{i=1}^{k} D_i.2 & \text{(since } f \text{ sends 2 units of flow from } s_i \text{ to } t_i) \\ & = & 2. \sum_{i=1}^{k} D_i & \end{array}$$

Thus $\widehat{C}(f) \geq 2.C(f^N)$. ■ **(Subclaim 1)**

Subclaim 2. *If f is any flow, then:*

$$\widehat{C}(f) \leq C(f) + C(f^N)$$

Proof.

Consider any flow f.

Consider any edge $e \in E$.

Case. $f(e) > f^N(e)$.

$$\begin{array}{lll} & [\widehat{d}_e(f(e)) - d_e(f(e))].f(e) & \\ = & [d_e(f(e)) - d_e(f(e))].f(e) & \text{(by definition of } \widehat{d}_e\text{)} \\ = & 0 & \\ \leq & d_e(f^N(e)).f^N(e) & \text{(since } d_e \text{ non-negative)} \end{array}$$

Case. $f(e) \leq f^N(e)$.

$$\begin{array}{lll} & [\widehat{d}_e(f(e)) - d_e(f(e))].f(e) & \\ = & [d_e(f^N(e)) - d_e(f(e))].f(e) & \text{(by definition of } \widehat{d}_e\text{)} \\ \leq & [d_e(f^N(e)) - d_e(f(e))].f^N(e) & \text{(since } f(e) \leq f^N(e) \text{ in this case)} \\ \leq & d_e(f^N(e)).f^N(e) & \text{(since } d_e \text{ non-negative)} \end{array}$$

In both cases we see that for every edge $e \in E$, we have:

$$[\widehat{d}_e(f(e)) - d_e(f(e))].f(e) \;\leq\; d_e(f^N(e)).f^N(e) \qquad \textbf{(2.1)}$$

Now consider the difference of the cost functions for flow f:

$$\begin{array}{lll} & \widehat{C}(f) - C(f) & \\ = & \sum_{e\in E} \widehat{d}_e(f(e)).f(e) - \sum_{e\in E} d_e(f(e)).f(e) & \text{(by definitions of } C \text{ and } \widehat{C}\text{)} \\ = & \sum_{e\in E}[\widehat{d}_e(f(e)) - d_e(f(e))].f(e) & \\ \leq & \sum_{e\in E} d_e(f^N(e)).f^N(e) & \text{(by } \textbf{(2.1)}\text{)} \\ = & C(f^N) & \text{(by definition of } C\text{)} \end{array}$$

Thus $\widehat{C}(f) \leq C(f) + C(f^N)$. ■ **(Subclaim 2)**

Proof of Theorem (contd).

From Subclaims 1 & 2, we get:

$$2.C(f^N) \leq \widehat{C}(f^*) \leq C(f^*) + C(f^N)$$

Thus $C(f^N) \leq C(f^*)$. ■ **(Theorem)**

Corollary. *Suppose that all the delays are queuing delay functions. Let f^N be a Nash flow. Let f^* be an optimal flow for twice the original capacity. Then:*

$$C(f^N) \leq C(f^*)$$

CS 684 Algorithmic Game Theory	**Scribe: Christina Tavoularis**
Instructor: Eva Tardos	**Monday February 16th, 2004**

1 Brower Fixed Point Theorem

The Brower fixed point theorem is the main tool for showing the existence of many problems in game theory for which polynomial time algorithms have not been found, and hence is an important result. The theorem can be proved using Sperner's Coloring Lemma for simplices of arbitrary dimension. In today's lecture, we will state the theorem, then prove a strong version of the coloring lemma before proving the theorem.

We will begin with a statement of the theorem. Consider the area in a k-dimensional simplex, with $k+1$ vertices such that $S_k = \{x \doteq \sum \lambda_i e_i : \lambda \geq 0 \text{ and } \sum \lambda_i = 1\} \subseteq \Re^k$, where $e_i = (0, ..., 0, 1, 0, ..., 0)$ is the unit vector in the i^{th} direction.

Theorem 1 (Brower Fixed Point Theorem) *If the function $f : S_k \to S_k$ is continuous, then there is $x \in S_k$ such that $f(x) = x$.*

For simplicity and illustrative purposes, we will consider a two-dimensional simplex ($k = 2$). Subdivide the simplex in every dimension and color the vertices with 3 colors, such that each corner has a different color. The vertices on the straight line between two corners can only be colored by the colors of the corners. A 3-colored $\triangle$ refers to a small triangle in the divided simplex such that each of its vertices is a different color.

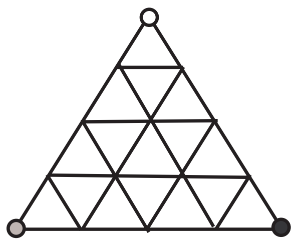


Figure 1: Subdivided 2-dimensional simplex such that each corner has a different color.

Lemma 2 (Sperner's Lemma) *3-colored $\triangle$s exist.*

In the figures, we have chosen white (w), grey (g) and black (b) to color the vertices in the divided simplex. Consider a walk on $\triangle$s that starts on the g-w side of the simplex and crosses over g-w edges only. Note that this walk will not form cycles because $\triangle$s have 3 edges and there are at most two g-w edges in a $\triangle$ (one to enter and one to exit). There are two options to terminate this walk. For option 1, the walk leads to a 3-colored $\triangle$, thus confirming existence. In option 2, the walk leaves the simplex on a g-w edge. A key observation is that the number of g-w edges on the g-w side of the simplex is odd. We would like to make use of the following Lemma which is stronger than Sperner's Lemma.

Lemma 3 *The number of 3-colored triangles is odd (this fact is needed for induction).*

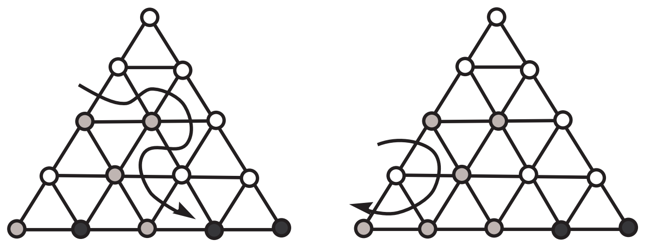


Figure 2: An example of a walk that ends at a 3-colored △ and an example of a walk that leaves the simplex.

Proof (of Sperner's Lemma) The walk pairs up g-w edges on the side or else leads to a 3-colored △. The fact that their are an odd number of edges on the g-w side of the simplex implies that at least one 3-colored △ exists. ■

Proof (of lemma 3) As shown in the proof of Sperner's lemma, an odd number of walks lead to 3-colored △s. We know that all walks lead to different 3-colored △s, because there is exactly one g-w edge for a walk to enter from in a 3-colored △. An important question to ask is if all 3-colored △s are lead to by such walks? This is false! A walk terminates once it has reached a 3-colored △. However, a reverse walk starting from a 3-colored △ and crossing only g-w edges can lead to another 3-colored △ or else to a g-w edge on the g-w side of the simplex. However, these adjacent 3-colored △s pair up (that is, there is an even number of them), and hence the total number of 3-colored △s is in fact odd. ■

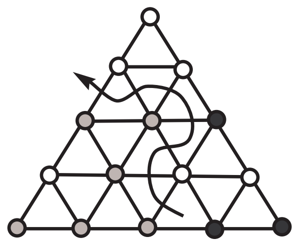


Figure 3: An example of a backwards walk that passes through multiple 3-colored △s.

Claim The proof of Sperner's Lemma generalizes to higher dimensions.

This will not be proved here, however the idea is very much the same as for $k = 2$. Consider a side of the simplex. This side uses k colors, and so we can use the k dimensional version of the theorem. We get by the induction hypothesis that there are an odd number of k colored simplices on this side. Now we consider a walk through such k-colored simplices using exactly the given k colors. We claim as was done above that such walks pair up the k-colored side simplices, or lead to $k + 1$ colored simplex in the larger dimension.

Proof (of Fixed Point Theorem) To start with we will color all nodes $x \in S_k$. Assume that $x = \sum \lambda_i e_i$ and $f(x) = \sum \mu_i e_i$. Assign x color i if $\lambda_i > \mu_i$ and this is the smallest such index (Note: if no such i exists, then $f(x) = x$ and we are done).

Now we want to use the Sperner lemma. Note first that the vertex e_i gets color i (as $\lambda_i = 1$, and all other values are 0). Furthermore, no points on a side of the simplex are colored by the opposite vertex. To see this note that a point x on the side opposite to vertex i has $\lambda_i = 0$, and hence cannot receive color i.

To use the Sperner lemma we need a subdivision of the simplex. We will consider a sequence of subdivisions. In the nth subdivision, each side of the simplex is subdivided into n segments. By the Sperner lemma, this subdivision has a multicolored simplex. Let x_n be the center of such a simplex. We want to argue that this sequence of points $x_1, x_2, \ldots$ has a limiting point that is a fixed point. First note that the x_i's are selected from a bounded and closed set (the simplex), so a subsequence of the sequence has a limit. Let x be such a limit.

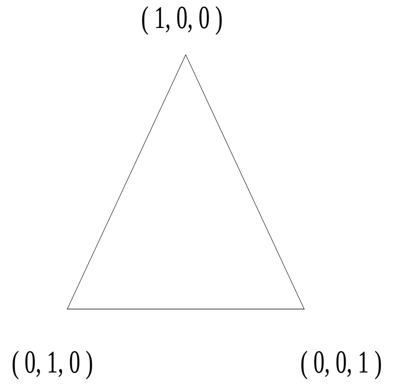
We claim that $f(x) = x$. Suppose this is not the case, and let $x = \sum \lambda_i e_i$, and $f(x) = \sum \mu_i e_i$. If $f(x) \neq x$ then there is an index j such that $\mu_j > \lambda_j$. Let $\epsilon = \mu_j - \lambda_j$. For all points x' in a small enough neighborhood of x we must have $x' = \sum \lambda'_i e_i$ with $\lambda'_j < \lambda_j + \epsilon$. Furthermore, as f is continuous, for all x' in a small enough neighborhood of x we must have that $f(x') = \sum \mu'_i e_i$ with $\mu'_j > \mu_j - \epsilon$. Let δ be the smaller of the two neighborhoods above. We get that in the δ neighborhood of x there is no point colored j. However, this leads to a contradiction: The point x is the limit of centers of increasingly small simplices, and thus for high enough n in the sequence if x_n is in the subsequence its whole simplex is in the δ neighborhood of x. This is a contradiction, and therefore the simplex contains all colors.

Now consider multicolored simplices on finer and finer subdivisions. f is continuous and some subsequence has limit x. The claim is that $f(x) = x$. If x has a neighborhood on which this coordinate increases, $f(x)$ has coordinate i increased. The nodes on top of it want to have coordinates pushed down but the sum has to be 1 so that can't happen. As n gets bigger $x_1, x_2, ..., x_n, ...$ the triangulation gets smaller. ■

CS 684 Algorithmic Game Theory	**Scribe: Christina Peraki**
Instructor: Eva Tardos	**February 18, 2004**

Last time we talked about the Brower fixed point theorem. The goal now is to use this in a game theoretic concept. Let $S_k = \{x = \sum_i \lambda_i e_i, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$, where e_i is the unit vector in coordinate i direction.

Theorem 1 *Let $f : S_k \rightarrow S_k$ be a continuous function. Then, there is $x \in S_k$ such that $f(x) = x$.*

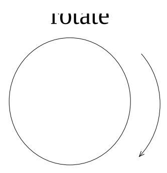


Extension: More generally, (that is, if we want to have some other region), let $f : R \rightarrow R$ be a continuous function. Then, there is $x \in R$ such that $f(x) = x$, assuming R is closed, convex and bounded.

This extension is very easy to prove if the convex set can be written in the form of S_k, that is, as a convex combination of a finite set of points. It is also true if instead of a finite set of points, we take an infinite set of points but in a bounded region.
Let us look at the following bad examples when R is either not closed, not convex or not bounded:

1. *not bounded:* Consider a line, which is a closed, convex, but not bounded region of the plane. What is wrong with the line? We can always shift, so there is no fixed point, as is shown in Fig. 1:

2. *not convex:* Consider a circle, which is a closed, bounded, but not convex region of the plane. What is wrong with the circle? We can always rotate, so, again, there is no fixed point, as shown in Fig. 2:



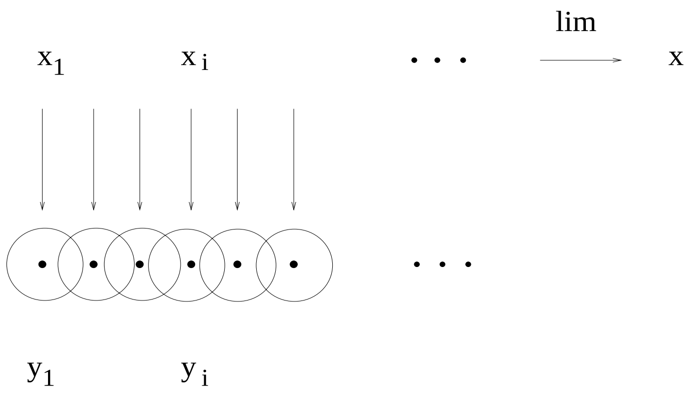
3. *not closed:* Consider an interval, which is a convex, bounded but not closed region. What is wrong with the interval? We cannot have a fixed point, again, by shifting less and less, e.g. $x \rightarrow \frac{x}{2} \rightarrow ....$

We will use the extension of the fixed point theorem for our game theoretic applications:

Theorem 2 Kakutani fixed point theorem: *Let R be closed, bounded and convex. Let $f : R \rightarrow \{subsets\ of\ R\}$ be a continuous function (for any point x, $f(x)$ is not a point but a region). Then, there is $x \in R$ such that $x \in f(x)$ (now $f(x)$ is a set).*

What does it mean for a set function to be continuous?

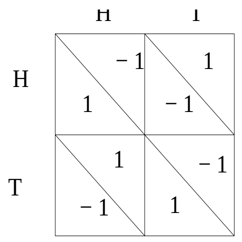
Let $x_1, x_2, ...$ be a sequence with $\lim x_i = x$. If $y_i \in f(x_i)$ (y_i is the best response for x_i), and $\lim y_i$ exists, then $y = \lim y_i \in f(x)$. (weak notion of continuity):



Goal: Does Nash equilibrium exist?

Suppose there are n players and each player i has a fixed set of possible strategies S_i.

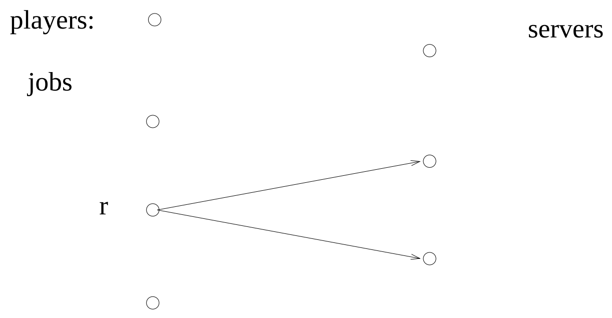
Example: *Matching pennies:* In this case, what is given is the payoff$_i(s_1, ..., s_n) \in R$ for all


H
T
H
T
-1
1
1
-1
1
-1
-1
1

$s_j \in S_j$, and player i. s_1 in this function is a permitted strategy for player 1, s_2 is a permitted strategy for player 2, etc.

Load balancing: The players in this game are the n jobs and each job i is allowed to go to a set $S_i \in S$ of servers, where S is the set of all servers.

Let us consider mixed strategies. A **mixed strategy** is a randomized strategy, that is, a mixed strategy for player j is a probability vector: probabilities p_{js} for all $s \in S_j$ with $p_{js} \geq 0$ and $\sum_{s \in S_j} p_{js} = 1$.



We care about the expected payoff for player i (0 if $p = \frac{1}{2}$ to win and $p = \frac{1}{2}$ to lose):

$$\text{payoff}_i(p_1, ..., p_n) = \text{Expected}(\text{payoff}_i | \text{ strategies chosen } \textbf{independently} \text{ with given probability}),$$

assuming p_j is a probability distribution.

The set of mixed strategies is *Nash* if for all players i and all probabilities p'_i we have:

$$\text{payoff}_i(p_1, ..., p_n) \geq \text{payoff}_i(p_1, ..., p_{i-1}, p'_i, p_{i+1}, ..., p_n),$$

that is, if player i changes to other probabilities, the payoff will not be better.

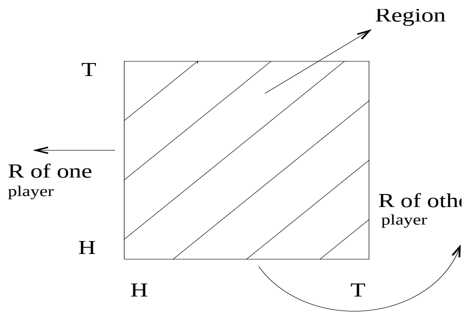
Theorem 3 *(Nash) Nash exists.*

Proof: (The space we are at is precisely the simplex, the probability distribution). Let R be the set of all mixed strategies, that is:

$$R = \{p_{js} \text{ for all } j, \text{ and } s \in S_j, \text{ such that } p_{js} \geq 0, \sum_s p_{js} = 1 \text{ for all } j\}.$$

Goal: Set $f(p_1, ..., p_n) = \{(p'_1, ..., p'_n)$ such that p'_j is the best response for player $j\}$.
Idea: p'_j : player j's best move if other players play $p_1, ..., p_n$.

Example: For matching pennies we have the following figure:



We skatch two proofs. First using Kakutani's fixed point theorem, and then a different proof using only Brouwer's fixed point theorem.

Idea: Apply Kakutani fixed point theorem: There is some $p_1, ..., p_n$ such that $(p_1, ..., p_n) \in f(p_1, ..., p_n)$. (This is exactly what we're looking for! That is the player's best choice).

Is this f continuous? According to the definition given earlier, it is. This finishes the first proof.

Next we want to sketch a proof using only Brouwer's fixed point theorem. For this we need to find p'_{js} for $s \in S_j$ in order to define $f(p_1, ..., p_n) = (p'_1, ..., p'_n)$, so that f is continous, and a fixed point is the equlibrium we are looking for. The idea is to consider,

$$d_{js} = \text{payoff}_j(p_1, ..., p_{i-1}, s, p_{i+1}, ..., p_n) - \text{payoff}_j(p_1, ..., p_n),$$

where $\text{payoff}_j(p_1, ..., p_{i-1}, s, p_{i+1}, ..., p_n)$ is the payoff for switching to the deterministic strategy s.

Let us take the following steps:

- It would seem natural to define p' as follows: $p'_{js} = p_{js} + d_{js}$, where p_{js} is increased if s is better and decreased if worse. However, what if d_{js} is a large negative number?
- To avoid this problem, let $p'_{js} = p_{js} + max(d_{js}, 0)$, where we changed the previous expression because if d_{js} is a large negative number we will no longer get a probability. However, there is still a problem, these "probabilities" do not sum up to one.
- So finally, we define $p'_{js} = \frac{p_{js}+max(d_{js},0)}{\sum_s p_{js}+max(d_{js},0)}$, where we normalized the previous expression to finally get a probability. This is also a continuous function.

In this way, f is continuous, so fixed point exists. Fixed point is Nash.

CS 684 Algorithmic Game Theory	**Scribe: Sumit Kunnumkal**
Instructor: Eva Tardos	**Friday, Feb 20, 2004**

1 Correlated equilibria

In this lecture we will look at correlated equilibria. We start with an example. Consider the game *Battle of Sexes*. Recall, that this game has two players who want to go either to a baseball game (B) or a softball game (S). The payoff matrix is as given below. The first entry in a cell denotes the player I's reward and the second entry denotes player II's reward. We can see that (S,S) and (B,B) constitute deterministic Nashes, with rewards (2,1) and (1,2), respectively. A randomized Nash consists of player I choosing S with probability 2/3 and B with probability 1/3 and player II choosing S with probability 1/3 and B with probability 2/3. The expected reward in this case is (2/3, 2/3), which is fair but lower than the worst outcomes in the deterministic Nashes.

	S	B
S	2,1	0,0
B	0,0	1,2

Now, consider a situation where a "trusted" authority flips a fair coin and based on the outcome of the coin toss, tells the players what they should do. So, for example, if the coin shows heads, player I is told to choose baseball and player II is told to choose baseball. Similarly, both players are told to choose softball when the outcome is tails. **(??)** It is important to note that no individual party has an incentive to deviate from what they are told to do. In this case, when player I is told to choose B, he knows that player II is told to choose B as well. So, player I has no incentive to deviate and switch to S as the payoff would be lower (0 compared to 1). The advantage of following such a procedure is that the expected rewards are now higher – (3/2, 3/2) compared to that of (2/3, 2/3) from the mixed NE.

Outcome	Probability
(B,B)	1/2
(S,S)	1/2

A more interesting example of correlated equilibria is the game of *Chicken*. This is again a two player game with the payoff matrix as shown below. In this case, the worst outcome occurs when both players dare (D,D). The deterministic Nashes are (D,C) and (C,D) with rewards (7,2) and (2,7), respectively. **(??)** A randomized Nash has players I and II choosing C and D with probabilities 2/3 and 1/3, respectively. This randomized Nash has an expected reward of $4\frac{2}{3}$ for each player.

	C	D
C	6,6	2,7
D	7,2	0,0

Now, let's look what happens in the case of a correlated equilibrium. As before, a trusted party tells each player what to do based on the outcome of the following experiment:

Outcome	Probability
(C,D)	1/3
(D,C)	1/3
(D,D)	1/3

We note once again that the trusted party only tells each player what he/she is supposed to do. The trusted party does not reveal what the other player is supposed to do. It is a correlated equilibrium if no player wants to deviate from the trusted party's instruction.

(??) So, in the *Chicken* example, if the trusted party tells player II to dare, then II has no incentive to deviate. This is because II knows that the outcome must have been (C,D) and that player I will obey the instruction to chicken. Next, let us consider the case when player II is told to chicken. Then player II knows that the outcome must have been either (D,C) or (C,C), each happening with equal probability. II's expected payoff on playing C conditioned on the fact that II is told to chicken is $\frac{1}{2} * 6 + \frac{1}{2} * 2 = 4$. In the above expression, 6 is the payoff from I also playing C, i.e., the outcome was (C,C) and 2 is the payoff II gets when I plays D, i.e., outcome was (D,C). If player II decides to deviate, i.e., play D when told to play C, then the expected payoff is $\frac{1}{2} * 7 + \frac{1}{2} * 0 = 3.5 < 4$. So, the expected payoff on deviating is lower than the payoff on obeying the instruction of the trusted party. Therefore player II doesn't deviate. Since the game is symmetric, player I also has no incentive to deviate from the instruction of the trusted party. Note that in the case of the correlated equilibrium, the expected reward for each player is $= \frac{1}{3} * 7 + \frac{2}{3} * 4 = 5$. This is higher than the expected reward of $4\frac{2}{3}$ in the randomized Nash. Therefore, rewards can be made better by correlation.

(??) Finally , we consider the general case of k player matrix games, where player i has n_i pure strategies. Then, we can find a correlated equilibrium in time polynomial in $n_1 n_2 \dots n_k$ using linear programming.

Remarks

- We consider only atomic games, so that the number of strategies is finite.
- We contrast this with the problem of finding a Nash equilibrium for a general game, for which no polynomial time algorithm is known.

(??) For a correlated equilibrium, we need to find a probability distribution on the set of all possible strategies. Let s_i be an element of the set of pure strategies of player i and let $s = (s_1, \dots, s_k)$. Also let $\text{payoff}_i(s)$ be the payoff to player i when strategy s is followed by the players. $p(s_1, \dots, s_k)$ denotes the probability with which the trusted party observes the event $(s_1, \dots, s_k)$, in which case players $1, \dots, k$ are told to play strategies $s_1, \dots, s_k$, respectively. To ensure that a correlated equilibrium results, no player should have an incentive to deviate from the instruction. So, if player i is told to play $\overline{s_i}$, then then all other strategies for that player should have no better outcome. Thus we want the expected payoff of strategy $\overline{s_i}$ to be at least as great as the expected payoff when player i alone switches to some other strategy $s_i^{'}$.

$$\sum_{s:s_i=\overline{s_i}} p(s)\text{payoff}_i(s) \geq \sum_{s:s_i=s_i^{'}} p(s)\text{payoff}_i(s_1, .., s_i^{'}, ..s_k) \ \forall s_i^{'} \tag{1}$$

In addition, since $p(s)$ is a probability distribution, $p(s) \geq 0$ and $\sum p(s) = 1$. The above inequalities define a linear program which can be solved to get a correlated equilibrium. We note:

- The number of variables is $n_1 n_2 \dots n_k$.
- We have to divide both sides of (1) by $\sum_{s:s_i=\overline{s_i}} p(s)$ so that it represents the expected payoffs.

- If we want to find the "best" correlated equilibrium, then we can simply introduce an appropriate objective function in the LP. For example, if the objective function is social welfare, the objective would be to maximize $\sum_s p(s) \sum_i \text{payoff}_i(s)$

We conclude with the following facts:

Fact 1 *All mixed Nashes are correlated, so correlated equilibria exist.*

Fact 2 *All convex combinations of mixed Nashes are also correlated.*

CS 684 Algorithmic Game Theory	**Scribe: Mateo Restrepo**
Instructor: Eva Tardos	**February 23, 2004**

Comparing Nash equilibria to Optimal Solutions in the Load Balancing Game with randomized strategies.

Abstract: A randomized version of the (atomic) load balancing game is presented. Some similarities and differences with the continuous version of the same game are commented. The concept of Nash equilibrium for this game is explicit-ed in terms of the expected load experienced by jobs. It is proven that the maximum expected load experienced by any user in a Nash equilibrium is at most two times the minimum possible.

1 The Load Balancing Game — Randomized version

Recall from the first lectures the setting of the load balancing game. In this game we have a set of n parties, so called *jobs*, which we will denote by j, and a set S containing m *servers* or machines, that will be denoted by i.

As before each job j has an associated subset of servers $S_j \subset S$ to which it can access. However, for the main result of this lecture it will be assumed that $S_j = S$ for all j.

Each job j has a *weight*[1] w_j, with which it is going to load *one* of the servers. So we are dealing here with an *atomic* version of the load balancing game with one major modification. We will allow jobs to flip coins to decide to which server to go. In other words we will be considering randomized strategies instead of pure ones. Formally, each job j decides (independently of the others) a probability distribution between the machines in S_j. For convenience we think of this as a vector

$$\mathbf{p}^j = (p_1^j, p_2^j, \dots, p_m^j)$$

where p_i^j is the probability that job j goes *as a whole* to machine i. Clearly, we must require $\sum_{i=1}^m p_i^j = 1$, for every job j, and $p_k^j = 0$ when $k \notin S_j$.

Finally, each server has a *response time* function $r_i(L)$, that is a function of the total load on it. By the total load L_i we simply mean the sum of the weights of the jobs on it. For this lecture and the following we will assume

$$r_i(x) = x, \quad \text{(this lecture and the following)}$$

The fact that the jobs can now choose their strategies from a continuum, in a sense splitting their weights between different machines, can let one think that this game is identical as the corresponding continuous version of the load balancing game. As we will show this is not exactly true. A significant difference will be apparent when we explicitly describe what the goal of the game is and how a Nash equilibrium for it looks like.

[1]This weight is conceptually the same as our former processing time. We have changed the terminology just for clarity in the notation.

In this game each job wants to maximize its own welfare in a selfish manner. The goal of each job stated in English it would be: "to be on a machine with small load (response time)."

Since there is randomness involved we must be a little careful when stating mathematically what this means. We will assume each player is *risk neutral.* This means that its *payoff* is just the expected load on the machine it is on.

By *expected load on machine i* we mean, of course,

$$L_i = \sum_j p_i^j w_j.$$

It is tempting to guess that the expected load experienced by job j is $c'_j = \sum_i p_i^j L_i$.

Despite the appealing simplicity of this expression, it does not correspond to reality. Its major fault consists in not taking into account the fact that, once a job chooses a machine, its full weight is completely there. This can be more clearly seen with an example. Suppose there is only one job with weight $w_1 = 1$ that chooses to go to each of the m machines with equal probability $p_i^1 = \frac{1}{m}$. We then have $L_i = \frac{1}{m} \cdot 1 = \frac{1}{m}$ and the expression above would give $c'_j = \sum_i p_i^j \frac{1}{m} = \frac{1}{m}$. On the other hand the load experienced by job j is 1 in all situations, because "it is always in the machine it went to". Thus, the expected load experience by j cannot be given by c'_j.

A few moments of thought are enough to realize that the correct expression for the *expected load experienced by job j* is

$$c_j = \sum_i p_i^j \left(w_j + \sum_{k \neq j} p_i^k w_k \right) = \sum_i p_i^j \left(L_i + (1 - p_i^j) w_j \right) \tag{1}$$

The expression in parenthesis is nothing more than the expected load of machine i conditioning on the event that job j goes there. Summing overall machines i with the corresponding probability of j going to machine i gives the desired expected load experienced by j. The last term in the last expression corresponds to the correction to the expected load L_i that originates from conditioning j on itself.

2 Nash Equilibria

From the last comments of the preceding section it is easy to infer that a set of probability distributions (mixed strategies) form a *Nash equilibrium* if and only if for any job j and any machine machine i, for which $p_i^j > 0$, the expected load of this machine, assuming j goes to i, is at most the expected load of any other machine s, assuming j goes there. Formally:

A set $\{\mathbf{p}^j\}_j$ of probability distributions is a Nash equilibrium if and only if for all jobs j and all machines $i, s \in S_j$ we have the implication

$$p_i^j > 0 \quad \Rightarrow \quad L_i + (1 - p_i^j) w_j \leq L_s + (1 - p_s^j) w_j. \tag{2}$$

An important point here is that for any i the expression $L_i + (1 - p_i^j) w_j$ does not depend on $\mathbf{p}^j$, as can be easily checked from the definition of L_i. In words, the probability distribution belonging to job j has no effect on the expected load experienced by j on a particular server i. Thus, the changing of the probabilities $\mathbf{p}^j$ will not affect any of the expected experienced loads of j. It will only affect other job's experiences.

But does a Nash equilibrium exist? Sure, this follows immediately from the general theorem that assures the existence of Nash equilibria for games with a finite number of pure strategies.

Last, we note that any deterministic Nash is also a randomized Nash in which for each job one of the probabilities is set to 1 and the rest to 0. This can be checked directly from the characterization (2).

3 Measuring the quality of a solution

Given that c_j measures the cost of a particular solution to player j, the following are some of the natural measures of the social cost that would allow us to define a social optimum.

1. $\sum_j c_j$ (average social cost)
2. $\max_j c_j$

In what follows we are going to compare the quality of the worst Nash to that of the optimal solution with respect to the second measure $\max_j c_j$, for the special case in which $S_j = S$ for all j. In fore-coming lectures we are going to analyze still a third measure of quality which consists in taking the expected value of the maximum load on any machine.

4 Comparing the quality of the worst Nash to that of the optimal solution with respect to $\max_j c_j$

Recall that in the first lecture we had proved for the atomic version of the load balancing game with deterministic (pure) strategies, that the maximum load on any machine in a Nash equilibrium is at most twice the minimum possible. Here we prove an analogous statement for the present game by essentially copying the proof of the old result.

Theorem 1 *Let c_j denote the expected load experienced by job j in any Nash equilibrium and let c_j^* denote the same quantity for the best possible random assignment in the load balancing game. If $S_j = S$, then*

$$\max_j c_j \leq 2 \cdot \max_j c_j^*.$$

Proof. Set $OPT := \max_j c_j^*$. Noticing from (1) that $c_j^* = w_j + \sum_i p_i^{*j} \left(\sum_{k \neq j} p_i^{*k} w_k \right) \geq w_j$ we conclude get

$$OPT \geq \max w_j \tag{3}$$

we argue now that

$$OPT \geq \frac{1}{m} \sum_j w_j \tag{4}$$

To see this recall that, by definition, we have $c_j^* \geq \sum_i p_i^j L_i^*$, in the optimal solution. Multiplying these inequalities by w_j and then adding them up gives

$$\begin{aligned}
\sum w_j c_j^* &= \sum_j \sum_i w_j p_i^j L_i^* \\
&= \sum_i \left(\sum_j w_j p_i^j \right) L_i^* \\
&= \sum_i L_i^{*2} \\
&\geq \frac{1}{m} (\sum_i L_i^*)^2 \qquad (5) \\
&= \frac{1}{m} (\sum_j w_j)^2,
\end{aligned}$$

where, in (5), we have used the *power mean- arithmetic mean* inequality $\sum_i L_i^{*2} \geq \frac{1}{m} \left(\sum_i L_i^*\right)^2$ [2] and then we have used the fact

$$\sum_i L_i = \sum_{i,j} p_i^j w_i = \sum_i w_i,$$

when summing first over j. Now, since the value of OPT is by definition the biggest experienced load among the c_j's, using the result just showed we can write

$$\sum_j w_j OPT \geq \sum_j w_j c_j \geq \frac{1}{m} \left(\sum_j w_j \right)^2$$

which gives (4) upon dividing by $\sum_j w_j$ on both sides.

Now denote by J the machine with worst expected experience load in the Nash, that is, $c_J = \max_j c_j$. We know, from the characterization of Nash equilibria, that

$$c_J \leq L_s + (1 - p_s^J) w_J$$

for all other servers $s \in S\,(= S_J)$. Summing over all s we get

$$m c_J \leq \sum_s \left[L_s + (1 - p_s^J) w_J \right] = \left(\sum_{k=1}^n w_k \right) + (m-1) w_J \leq \sum_k w_k + m w_J$$

After dividing by m, the result now follows from (3) and (4)

$$c_J \leq \frac{1}{m} \sum_k w_k + w_J \leq OPT + OPT.$$

[2]This inequality is a special case of Jensen's inequality applied to the convex function $f(x) = x^2$, but has also a quick independent proof. We start with the generic inequality $0 \leq \sum_i (L_i^* + t)^2 = \sum_i L_i^{*2} + 2 \left(\sum L_i^* \right) t + m t^2$ that holds for any real t. Now putting $t = -(\frac{1}{m} \sum_i L_i^*)$ gives $\sum_i L_i^{*2} - \frac{1}{m} \left(\sum L_i^* \right)^2 \geq 0$ which easily yields the desired inequality.

■

The moral of the story is that deterministic Nashes are not much worse than randomized ones. Also, if one wants to prove something about Nash equilibria in general a first step should be to prove it for deterministic Nashes, which are special cases of the former and are also easier to work with. After that one might think of translating the proof for the case of mixed strategies.

5 Complementary References

- E. Koutsoupias, C. H. Papadimitriou. *Worst-case equilibria*, Symposium on Theoretical Aspects of computer science (STACS) 99E.

CS 684 Algorithmic Game Theory **Scribe: Xin Qi**
Instructor: Eva Tardos **February 25, 2004**

In this lecture, we continue to consider the randomized load balancing game. We will evaluate the quality of randomized Nash equilibria with respect to the optimum, but for a different quality measure.

1 Setting of the problem

There are n jobs and m servers. Each job j has weight w_j, and an available subset S_j of servers to choose from. We will use p_i^j to denote the probability for job j to choose server i, so a solution to this game is a set $\{p_j\}_j$ of probability distributions. Each job wants to experience as small load as possible.

The expected load experienced by job j is described as:

$$c_j = \sum_i p_i^j (w_j + \sum_{k \neq j} p_i^k w_k) \tag{1}$$

in which the term $(w_j + \sum_{k \neq j} p_i^k w_k)$ is the load experienced by job j if it goes to server i.

Alternatively, we can use a random variable X_j to denote the load experienced by job j. Then expected load can be expressed as:

$$c_j = Exp(X_j) \tag{2}$$

2 Quality measures

The following are some natural measures of social cost that enable us comparing Nash equilibria with social optima.

1. $\sum_j c_j$
2. $\max_j c_j = \max_j Exp(X_j)$
3. $Exp(\max_j X_j)$

Before, we have concentrated on the second measure. Last time, we discussed that if $S_j = S$ $(\forall j)$ then $\max_j c_j \leq 2 \cdot opt$. In homework, we saw the a special case of pure game where $w_j = 1$ $(\forall j)$. In that case, we know $\max_j c_j \leq O(\log m) \cdot opt$. Actually, that bound also holds for jobs with non-uniform weights, and for randomized games.

In this lecture, we will talk about the third measure. Although it looks like the second one, where the order of max and Exp is exchanged, it is very different. There might be some instances that every job has a very small probability of experiencing large delay, but for any solution, there are always some jobs experiencing large delay. For this kind of instances, the second objective function is very small, while the third one has a large value.

3 An instance with bad Nash equilibrium

In this case, we have $S_j = S$ $(\forall j)$ and $w_j = 1$ $(\forall j)$. We also let $m = n$. There is only one deterministic Nash, in which each job chooses separate server. It is also the optimal solution to the second and the third measures. However, there are many randomized Nash equilibria.

Claim 1 *The solution that every job chooses all servers uniformly randomly, i.e.* $p_i^j = \frac{1}{m}$ $(\forall j, \forall i)$ *is a Nash equilibrium.*

Proof. It is not hard to see this, since every server seems to be equal. To be a little more formal, let us take a look at the load experienced by job j if it goes to server i:

$$w_j + \sum_{k \neq j} p_i^k w_k = w_j + \frac{1}{m} \sum_{k \neq j} w_k = 1 + \frac{n-1}{n} \tag{3}$$

So each server i appears the same to job j, and therefore job j will be indifferent to choose among servers, provided all the other jobs choose servers uniformly randomly. ■

From the above proof, we also know that

$$c_j = 1 + \frac{n-1}{n} < 2 \ \ (\forall j) \tag{4}$$

So the second quality measure, which is the maximum expected load, is good for this solution. However, the third quality measure, the expectation of maximum load, is very bad. According to the results of Balls & Bins problem, which we have learned in CS681, maximum load on a server is $O(\log n / \log\log n)$ with high probability.

The remaining part of this lecture will focus on the proof of the above result, using Chernoff bound.

Chernoff bound (the stronger version) There are n constant numbers $a_1, a_2, \ldots, a_n$, s.t. $0 \leq a_1, a_2, \ldots, a_n \leq 1$, and n random variables $y_1, y_2, \ldots, y_n$, s.t. $y_i \in \{0, 1\}$ $(\forall i)$. Let

$$Y = \sum_{i=1}^{n} y_i a_i \tag{5}$$

and

$$\mu \geq Exp(Y) = \sum_i a_i Pr(y_i = 1) \tag{6}$$

We have the following bound:

$$Pr(Y > (1+\delta)\mu) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^\mu \tag{7}$$

Bound on the quality of randomized Nash equilibria Our goal is to use the above Chernoff bound to prove the following result:

$$Exp(\max_j X_j) \leq O(\log m) \cdot opt \tag{8}$$

With the stronger version of Chernoff bound described above, we can slightly generalize our problem to the case of non-uniform weights of jobs. Suppose job j has weight w_j. We can assume w.l.o.g., that $w_j \leq 1$ $(\forall j)$. This is true as we can always divide all weights by some factor of at least $\max_j w_j$. Actually, we will scale by op, which is clearly at least $\max_j w_j$. From last lecture, we know that under this assumption,

$$Exp(y^i) \leq 2 \tag{9}$$

We have the following random variables:

$$y_j^i = \begin{cases} 1 & \text{if job } j \text{ is on server } i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Then the load of server i can be described as:

$$y^i = \sum_j y_j^i w_j \tag{11}$$

Claim 2 *For any server i, and some constant c,*

$$Pr(y^i > c \cdot \log m) \leq \frac{1}{m^2} \tag{12}$$

Why is claim 2 useful? Because it implies:

$$Pr(\max_i y^i > c \cdot \log m) \leq \sum_i Pr(y^i > c \cdot \log m) \leq \frac{1}{m} \tag{13}$$

and therefore

$$Exp(\max_i y^i) \leq \frac{m-1}{m}(\log m + \frac{1}{m}\sum_j w_j) \leq c \cdot \log m + 1 \tag{14}$$

Now we will give the proof of claim 2.

Proof. Let $\mu = 2$, since $Exp(y^i) \leq 2$, and let $1 + \delta = \log m$. We can then apply Chernoff bound to y^i:

$$\begin{aligned} Pr(y^i > 2\log m) &= Pr(y^i > (1+\delta)\mu) \\ &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu = \left(\frac{e^{\log m - 1}}{(\log m)^{\log m}}\right)^2 \\ &\approx \left(\frac{e^{\log m}}{(\log m)^{\log m}}\right)^2 \\ &= \left(\frac{m}{m^{\log\log m}}\right)^2 \\ &= \left(\frac{1}{m^{\log\log m - 1}}\right)^2 \end{aligned}$$

For m that is big enough, obviously we have $m^{\log\log m - 1} \leq m$, which implies

$$Pr(y^i > 2\log m) \leq \frac{1}{m^2} \tag{15}$$

which is claim 2.

■

CS 684 Algorithmic Game Theory	**Scribe: Charles Dietrich**
Instructor: Eva Tardos	**Friday, Feburary 27, 2004**

1 Stable equilibria

Today we will consider the stability of equilibria. Natural game play does not reach Nash equilibria. The artificial intelligence community has looked into good strategies for game play, and there has been interest in learning good strategies, which we will cover in the second half of this lecture.

The first question we will consider is this: if players are in an equilibrium, and one player changes his/her strategy, what happens to the equilibrium? Consider for example the *Battle of Sexes* presented in the first lecture:

	B	S
B	$2,1$	$0,0$
S	$0,0$	$1,2$

We noted that (B,B) and (S,S) were the pure Nash equilibria. But suppose that two players were in the (B,B) Nash equilibrium and one player decides to choose S. The other player must now choose S as its Nash strategy. The players are once again in a Nash equilibrium! However, for the *prisoner's dilemma*:

	C	D
C	$3,3$	$0,4$
D	$4,0$	$1,1$

The unique pure Nash equilibrium is (D,D). If one player changes his/her strategy to C, then the other player does not have any incentive to change their strategy.

We define an equilibrium to be strict if for all players their best response is unique. So the *Battle of Sexes* does not have a strict equilibrium, but the *prisoner's dilemma* does.

Now we consider modeling the evolution of populations in biology. There is a natural application of games to evolution. We can consider the payoff of a strategy as the reproductive success of a player.

For biological applications, it makes sense to consider symmetric games, e.g. games in which the payoff matrix is the same for all players. In 2 player games, these are represented by symmetric matrices as above in the *prisoner's dilemma*. The first game we will consider is *Dove & Hawk*. Here is our first version:

	D	H
D	$1/2,1/2$	$0,1$
H	$1,0$	$1/4,1/4$

The way it works is as follows. Suppose there is some food source, and player A and player B arrive at the food source. If two doves arrive, they peaceably share the food. If a dove and a hawk arrive, naturally the hawk gets the food. Finally, if two hawks arrive, they fight. Their reward will be less than the two doves. In this version, a pure Nash equilibrium is (H,H). In fact this is the strict equilibrium.

So the question concering biologists is: given a population consisting of p proportion of doves and 1-p hawks, is it better to play dove or hawk? Note that by changing the 1/4,1/4 in the payoff matrix we can change the social optimum or even what the Nash equilibria are. One might imagine in some payoff matrices, the best strategy depends on the proportion p.

We define a strategy to be an Evolutionary Stable Strategy (ESS) as: if everyone plays stable strategies and ε percentage plays some other strategy, then there is a small enough $\varepsilon > 0$ so that those playing the this alternate strategy get a smaller reward.

Consider the following variation of the *Dove & Hawk* game above:

	D	H
D	$1/2, 1/2$	$0, 1$
H	$1, 0$	$0, 0$

Now (H,H) is still a Nash equilibrium. But it is not a strict equilibrium, since there is no penalty for playing dove when the other player plays H. It is an ESS though: Suppose $\varepsilon > 0$ plays D

- for doves D
  - $1 - \varepsilon$ meet H, get a reward of 0
  - ε meet D, get a reward of 1/2
- for hawks H
  - $1 - \varepsilon$ meet H, get a reward of 0
  - ε meet D, get a reward of 1

Now consider the following *Dove & Hawk* game:

	D	H
D	$1/2, 1/2$	$0, 1$
H	$1, 0$	$-1/2, -1/2$

The pure Nash equilibria are (H,D) (D,H). Mixing probabilities (1/2, 1/2) is also an equilibrium. This mixed equilibrium is evolutionarily stable. Thus a population of 1/2 hawks and 1/2 doves is stable.

Now we turn our attention to learning. Consider a game based on *prisoner's dilemma.* Two players meet in a match, and each chooses to either cooperate or defect. After they have chosen, the choices for each player are revealed, and the payoff for each player is calculated and added to each player's score. The players retain knowledge of prior games against this opponent. The artificial intelligence community has played a version of this game where there are a set of n players, and each player plays each other player k times, for some k. The goal is to maximize the sum of the rewards from each match.

Consider the following strategies. Suppose we knew that the other player would always defect. Then optimally we would also always choose to defect. On the other hand, if the other player always cooperated, we would optimally always choose to defect. So an obvious strategy is to always defect. But of course, if everyone does this then we score fewer points than if everyone had chosen to cooperate.

Consider another strategy, *tit-for-tat.* In this strategy, for each time an opponent defects, we defect once. Thus if we are paired with a pure defector, we only come out 1 point the worse over k matches. If we are paired with a pure cooperator, we do as well as a pure cooperation strategy. If we are paired with another *tit-for-tat* player, we also do as well as a pure cooperation strategy. Note that in the last strategy, we do better than if we had defected always, and in the first strategy we do better than if we had cooperated always. So *tit-for-tat* would seem to be a good strategy.

Suppose we had a population consisting of C, D and *tit-for-tat.* In the evolution game a population of *tit-for-tat* results. We can also show that tit-for-tat is an ESS, while neither C nor D is an ESS.

But if there are other variants, for example *tit-for-2-tat*, then some other strategy may win. In the games that the AI people play, other variants have shown superior to *tit-for-tat.*

Next week we will look at market equilibrium.

CS 684 Algorithmic Game Theory	**Scribe: Chandrashekhar Nagarajan**
Instructor: Eva Tardos	**1st March, 2004**

1 Market Equilibria

Market life is actually a potential game played by real persons with parameters similar to the potential game parameters. However the decisions taken by the players seek to maximize their own private utility. We will consider some simple market games in this lecture and try to analyze their equilibria if exists.

1.1 Simple example

Let us look at a very simple market game. This is a seller-location game where there is demand for only one product, say ice-cream. Consider a scenario where the demand is from the buyers who are uniformly distributed on a line segment from 0 to 1 on the real line $\mathbf{R}$. Also assume that the buyers will get their ice-creams from the nearest seller. We will analyze the game for k sellers where the objective of each seller is to place his shop on $[0, 1]$ so as to maximize his profit i.e the number of customers served by him.

Case 1. $k = 1$ This is the case of monopoly. The seller can place his shop anywhere on $[0, 1]$ and the buyers have no other options but to get their ice-creams from him. So any location is a Nash as he has no incentive to change his current location.

Case 2. $k = 2$ In this case there are two sellers who want locate their shops on $[0, 1]$ so as to maximize the number of customers served by each of them. We will discuss how the Nash solution in this case looks like if it exists. Suppose in a solution if the location of shops of two sellers are separated by some distance then this solution cannot be Nash as each seller can increase the number of customers by moving closer to the other seller. Also the solution where the two sellers are at same location other than 0.5 can't be a Nash as any seller can move closer to 0.5 to increase his number of customers. So the only Nash solution is when both sellers are at 0.5 on the line $[0, 1]$. So the only deterministic Nash is when the sellers are located on top of each other at 0.5

Case 3. $k = 3$ In this case where there are 3 sellers if all the 3 sellers are not located on each other then the single seller who is not in the middle of other sellers will tend to move toward other sellers to increase his share of customers. So there cannot be a deterministic Nash in this scenario. Even if all three sellers are top of each other at some location other than 0.5 then any one seller can move towards 0.5 and increase his share of customers from one-third to some fraction greater than half. If all customers are at 0.5 any single seller can move bit from 0.5 to increase his share from one-third to close to half of the customers. So in this case no deterministic Nash solution exists as we have exhausted all possibilities.

1.2 Same example with prices

In the previous example we considered the price of an ice-cream is the same irrespective of the seller. In this model we allow the sellers to determine their price for the product. One problem in this model is that the buyers may have a trade-off between cheapness of the product (ice-cream) and the nearness of its availability. Another problem is that the price may affect the quantity of ice creams bought by the buyer. We may also have to think about how to determine the social goodness in this model. One way is to consider social goodness as more goods produced for lesser price. So the seller will have to strike a balance between minimizing the price to attract the buyers and also maximize his profit.

1.3 A market game with prices (Vetta 2002)

Input :

- n markets (M).
- k suppliers.
- L_j, a set of possible locations for each supplier j.
- A bipartite graph on set of markets and set of possible locations with cost on edge $c_{ij} \geq 0$ which is the cost for serving market i from location j.
- p_i, maximum price that can be paid by market i to buy the product.

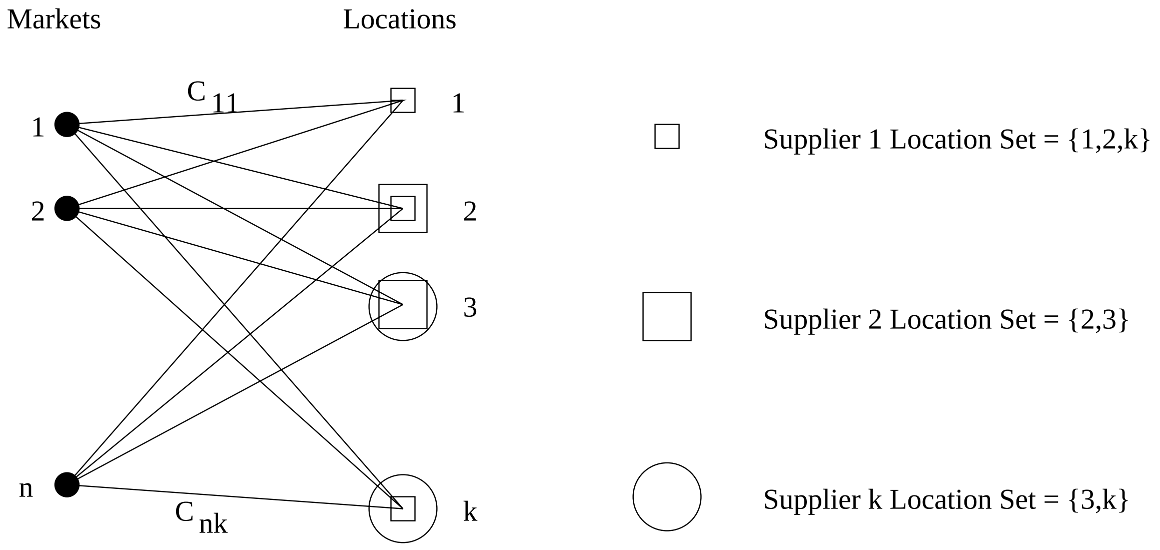

Markets
Locations
C 11
1
2
n
1
2
3
k
C nk
Supplier 1 Location Set = {1,2,k}
Supplier 2 Location Set = {2,3}
Supplier k Location Set = {3,k}

Figure 1: Market - Supplier game.

Let us consider a special case of this model where the locations of the suppliers are fixed. Let L' be the set of locations from which the markets can be served i.e the locations where suppliers are available. Let us assume that π_{ij} be the price determined by the supplier at location $j \in L'$ to the market i.

Let us analyze the game in the perspective of markets and the suppliers. The market i tries to get their goods from the cheapest available location $j \in L'$. Let us call this cheapest location $j = \sigma(i)$.

$$\sigma(i) = arg\left(\min_{j \in L'}(\pi_{ij})\right)$$

The supplier at a location $j \in L'$ tries to lower the price π_{ij} at market i below other locations price $\pi_{ij'}, j \neq j'; j' \in L'$ provided $\pi_{ij} \geq c_{ij}$. The profit he makes at market i is $\pi_{ij} - c_{ij}$. If $\pi_{ij} < c_{ij}$ then the supplier at j is at a loss selling goods at market i. A supplier at location j will be able to serve the market i over other locations in L' if and only if his price is the minimum of all prices the locations that serve the market i. Also note that to be able to lower the price below others and at the same time get profit, the supplier at location j have the cost c_{ij} less than other costs $c_{ij'} j' \in L'$. So

$$\sigma(i) = arg \left(\min_{j \in L'} (c_{ij}) \right)$$

The supplier at location $\sigma(i)$ will assign a price equal to second minimum of $c_{ij}; j \in L'$ or p_i whichever is smaller if $c_{i\sigma(i)} \leq p_i$[1]. So

$$\pi_{i\sigma(i)} = \min \left(p_i, \min_{\substack{j \in L' \\ j \neq \sigma(i)}} c_{ij} \right)$$

Theorem 1 *This game is a potential game.*

Proof : Let M' be the set of markets that are assigned to some location in L'. Let us define a function

$$\Phi = \sum_{i \in M'} c_{i\sigma(i)} + \sum_{i \notin M'} p_i$$

This potential function gives some idea on the quality of the solution. This potential function is actually the social value function which captures the actual cost of the situation. To show that this is indeed a potential function we have to show that a single selfish move of a particular supplier changes the potential function by exactly the amount of change in the supplier's revenue. Consider a supplier changing his location from j to j'. Let the potential function change from Φ to Φ'. Let $\sigma'(i)$ be the new locations associated with market i after the location change of a supplier from j to j'. We abuse the notation $c_{i\sigma'(i)}$ and $c_{i\sigma'(i)}$ to also denote p_i if the corresponding market i is not assigned at all to any location.

Divide the markets M into four disjoint sets S_1, S_2, S_3, S_4 defined as follows.

$$S_1 = \{i : \sigma(i) = j \textit{ and } \sigma'(i) = j'\}$$
$$S_2 = \{i : \sigma(i) = j \textit{ and } \sigma'(i) \neq j'\}$$
$$S_3 = \{i : \sigma(i) \neq j \textit{ and } \sigma'(i) = j'\}$$
$$S_4 = \{i : \sigma(i) \neq j \textit{ and } \sigma'(i) \neq j'\}$$

Consider the change in profit P_i of the supplier corresponding to each of these set of markets S_i.

$$P_1 = \sum_{i \in S_1} (c_{i\sigma(i)} - c_{i\sigma'(i)})$$

[1]If $c_{i\sigma i} > p_i$ then the supplier won't get any profit by selling to market i as the market will not pay a price $> p_i$

$$P_2 = -\sum_{i \in S_2} (c_{i\sigma'(i)} - c_{i\sigma(i)})$$

$$P_3 = \sum_{i \in S_3} (c_{i\sigma(i)} - c_{i\sigma'(i)})$$

$$P_4 = \sum_{i \in S_4} (c_{i\sigma(i)} - c_{i\sigma'(i)}) = 0$$

Each of the profit P_i is computed as the difference of the revenue supplier gets after and before the change of his location from j to j'. Profit from each of market i from S_1 is

$$(\pi_{ij'} - c_{i\sigma'(i)}) - (\pi_{ij} - c_{i\sigma(i)}) = c_{i\sigma(i)} - c_{i\sigma'(i)}$$

because $\pi_{ij'} = \pi_{ij}$ as the second minimum of the costs for the market i has not changed. Similarly for a market i in S_2 there has been a loss of revenue due to the shift. So the profit is

$$-(\pi_{ij} - c_{i\sigma(i)}) = -((c_{i\sigma'(i)} - c_{i\sigma(i)})$$

because $\pi_{ij} = c_{i\sigma'(i)}$ as the location corresponding to the second minimum among the costs is the new location which will be assigned to the market i. Similar arguments holds for P_3 and P_4

So the total profit

$$P = P_1 + P_2 + P_3 + P_4 = \sum_{i \in M} (c_{i\sigma(i)} - c_{i\sigma'(i)})$$

Also note that the change in potential function

$$\Phi - \Phi' = P$$

So this means that the decrease in the function Φ is exactly equal to the profit of the supplier changing his location. So this is a *Potential Function* by its definition. ■

Corollary 2 *There is a* Deterministic Nash *in this game.*

Proof : As this game is a potential game, a sequence of finite selfish moves leads to a Deterministic Nash. No state is repeated as the potential function Φ strictly decreases after each move leading to a Deterministic Nash. ■

Finally, let us consider the total value of a solution. If the market i is getting supplied by supplier at location $\sigma(i)$, then this contributes $p_i - c_{i\sigma(i)}$ to the total happiness. To see why, let $j = \sigma(i)$, and assume the prize is π_{ij}, then the customer i has benefited $p_i - \pi_{ij}$, while the suppier at j benefited $\pi_{ij} - c_{ij}$ from the sale, so the total benefit is $p_i - c_{i\sigma(i)}$ as claimed. If the market i is not being supplied, than there is no benefit. Using the potential function Φ we defined above, we get that the total benefit of the solution is exactly

$$\sum_i p_i - \Phi.$$

Theorem 3 *The social value of any Deterministic Nash in this game is at least 1/2 of the best possible social value of any solution.*

Proof : Assume that in a Nash solution providers locate at locations $j_1, \ldots j_k$ respectively, and in an optimal solution they locate at locations $j'_1, \ldots, j'_k$. Let $\sigma(i)$ denote the provider assigned to customer i in Nash, and $\sigma'(i)$ the provider assigned in the optimal solution. Now consider the possible move of a provider ℓ moving its location j_ℓ to j'_ℓ. We know from Theorem 1 that the benefit to prover ℓ of this move, is exactly the improvement of the potential function Φ, i.e., exactly the same as the improvement in the total benefit $\sum_i p_i - \Phi$. The solution is a Nash, so the change of moving a single provider does not improve his benefit, and so also does not improve the total benefit. Note, however, that it is possible that moving multiple providers improves the benefit, even if moving a single provider does not. Next we want to bound this improvement possible by the current total benefit of all providers, and this will prove the theorem.

Let $val'(\ell)$ be the new benefit of provider ℓ when he *alone* moves to location j'_ℓ. Consider a custimer i, and let $\delta(i)$ denote the difference between the cost associated with serving customer i in the two solutions. More formally, if i is being served by both solutions then let $\delta(i) = c_{i\sigma(i)} - c_{i\sigma'(i)}$, if i is not served by one of the two solutions, then replace the corresponding cost by p_i in this expression. Now let $I'(\ell)$ be the set of customers served by provider ℓ in the optimal solution. The key observation is that

$$val'(\ell) \geq \sum_{i \in I'(\ell)} \delta(i).$$

This holds with equality if all customers $i \in I'(\ell)$ are served by other providers in the Nash, and the inequality holds, if some customers stay with the same provider.

Now, let $val(\ell)$ be the value of provider ℓ in the Nash solution. This solution is a Nash, so by definition $val'(\ell) \leq val(\ell)$ for all providers ℓ. Adding these inequalities we get the following.

$$\sum_\ell val(\ell) \geq \sum_\ell val'(\ell) \geq \sum_\ell \sum_{i \in I'(\ell)} \delta(i) = \sum_i \delta(i).$$

The right hand side is exactly the improvement in the total improvement in value going from Nash to the optimal solution. Let Φ denote the potential at Nash, and Φ' denote the potential at the optimum. So we get the following bound on the value of the solution.

$$\sum_i p_i - \Phi' = (\sum_i p_i - \Phi) + (\Phi - \Phi') = (\sum_i p_i - \Phi) + \sum_i \delta(i) \leq (\sum_i p_i - \Phi) + \sum_\ell val(\ell) \leq 2(\sum_i p_i - \Phi),$$

where the last inequality follows as $\sum_\ell val(\ell)$ is the providers total value, while $\sum_i p_i - \Phi$ is the total value also including the customers. ■

CS 684 Algorithmic Game Theory	**Scribe: Ara Hayrapetyan**
Instructor: Eva Tardos	**March 3, 2004**

1 Market Equilibrium

Our next topic where we will spend a few lectures will be games related to market equilibria. In today's game we will have k players who bring n different kinds of goods to the market. The players obey a *pricing mechanism*, which states that given a price $p_i > 0$ for good i, the players will sell the items they brought and will purchase goods so as to maximize a personal utility.

The input to the problem is a vector $e_1, \dots, e_k \in R^n$, where each element e_i is itself a vector of quantities that player i brought to the market. We are also given utility functions $u_i(x) : R^n \mapsto R$ denoting the utility that a player i receives from purchasing quantities of goods described by the vector x. We assume that $u_i(x) > 0$ and that $u_i(x)$ is strictly monotone and strictly concave. We will give an intuition on these assumptions shortly.

The question we would like to answer is whether there are a set of market clearing prices. Formally we desire a set of prices $p = p_1, \dots, p_n$ for the goods such that if players sell their goods at these prices and then purchase (with the revenue made from the selling) a set of goods so as to maximize their personal utility, there will be no envy (no good will be demanded more than it is supplied) and the market will be cleared (no good will be supplied more than demanded).

Given a set of prices p player i gets $p \cdot e_i$ from selling his goods (the $\cdot$ denotes a dot product). Then the vector of goods x_i that he wants is the solution of the following system:

$$max \ \ u_i(x) \tag{1}$$
$$s.t. \ \ p \cdot x \leq p \cdot e_i \tag{2}$$

The set of prices p is then at an equilibrium if $\sum e_i = \sum x_i$.

A brief reason why it makes our job easier to assume strictly monotone utility functions is that the strictly monotone utility ensures that at price 0 everyone will want infinite quantities of the good, therefore the solution will produce strictly positive prices.

And the reason why we want strictly concave is that it makes the solution to the above system unique, since if x_1 and x_2 are two feasible optimal solutions then $\frac{x_1+x_2}{2}$ is a strictly better solution.

Now comes our main theorem.

Theorem 1 *Equilibrium prices do exist under the above assumptions.*

Proof: We want to prove this using the Kakutani fixed point theorem. So we have set up the set function. The overall idea is that if product j is over-produced its price should go down and if it is under-produced its price should go up. First, without loss of generality we can assume that $\sum p_i = 1$ (scale), so we have a simplex. Let z denote the excess in the market, i.e. $z = \sum x_i - \sum e_i$. Our intuition tells us that we want the price go up on the products whose component in z is positive, and down if it is negative.

Let $f(p) = \{q > 0 \mid \sum q_i = 1 \ \textit{and} \ q_j > 0 \ \textit{only if} \ j \ \textit{is a maximum component of} \ z\}$ for all $p > 0$. We claim that this produces the desired fix-point.

First notice that the prices p define f in an alternate way, i.e. $f(p) = \{q \mid qz \geq q'z \, \forall q' \; with \; \sum q_i = \sum q'_i = 1\}$. Unfortunately our function f defined thus far is not closed, since we defined it on strictly positive prices. We fix it by saying that for those price vectors p with at least one component equal to 0, the function f maps them to the set of vectors whose dot product with p is 0 (notice that we did this construction so as to ensure no fix-point among such sets since $p \cdot p$ is never 0).

If we can show that f is continuous, then we would be done, since as we just observed a fix-point p^* of f must have strictly positive prices, but then $p \in f(p)$ by definition puts price on every component, and thus every component has maximum excess therefore every component has excess 0.

We are left to show the continuity of f. Recall that for f to be continuous given a converging price sequence $p^1, p^2 \ldots \to p$ and a converging sequence of elements $q^i \in f(p^i)$ where q^i converges to q, we need to show that $q \in f(p)$. There are two cases to consider here, when $p > 0$ and when p has a zero component.

First consider the case when $p > 0$. For the p^i's the set of corresponding z^i's also clearly converges to z because z is a continuous function on prices and utilities and those are all continuous. Now we use the alternate definition of f. Since q^i is in $f(p^i)$, we have $q^i \cdot z^i \geq q' \cdot z^i$ for all q'. Again because of continuity, this inequality also holds in the limit as well, therefore $q \cdot z \geq q' \cdot z$ for all q', which means that $q \in f(p)$.

Due to lack of time, we will not discuss the case when p has a zero component in detail. For this case, the original proof of $f(p)$ seems more appropriate. We need to show that for every non-zero coordinate of q, the corresponding coordinate of p is zero. The idea is that once p^i has a small enough coordinate, than the demand in this product will be very high, and as a result, the maximum coordinates of z will be from among these products, so all other coordinates of q^i must be zero.

CS 684 Algorithmic Game Theory	Scribe: Lucian Leahu
Instructor: Eva Tardos	Friday, March 5, 2004

Market Equilibrium (cont'd)

Last time, we started analyzing equilibrium in markets. We have proved that equilibrium prices exist.

Today, we will consider a slightly different scenario:

- the utility function is linear
- goods are separated from buyers (last time the price was affected by how much money the players were making by selling their goods. Today we assume that there are b amounts of goods on the market. Each player i has m_i money.)

Setup

Consider the case with k buyers and n goods. Let $U_i(x)$ be the utility of buyer i:

$$U_i(x) = U^i \cdot x$$

Recall from last time that each buyer i maximizes $U^i \cdot x$, subject to $p \cdot x \leq m_i$ and $p > 0$. Therefore, the best goods for player i are:

$$\max_j \frac{U^i_j}{p_j}$$

Each player i only wants only wants goods j that are the best given the above definition.

We would like to find a combinatorial/algorithmic way to prove the existence of equilibrium prices.

Consider a bipartite graph consisting of buyers on one side and goods on the other side. We add an edge (i, j) if:

$$\max_l \frac{U^i_l}{p_l} \text{ is achieved for } l = j.$$

Therefore:

- all players' nodes have "outgoing edges" (i.e. there exists a max value for each player)
- it is possible that some nodes representing goods have no "incoming" edges.

Prices p are in equilibrium if for each player i, we have quantity x^i such that:

$$\begin{aligned} x^i_j &> 0 \Rightarrow (i,j) \in E \\ x &\geq 0 \text{ and all products and money are used up} \end{aligned}$$

Given prices, we can decide if they are equilibrium prices using the bipartite graph above and flows:

- buyers have supply m_i
- each good j has demand $\sum_{j=1}^{n} b_j p_j$ (we want to sell everything)

We will describe an algorithm that monotonically increases prices. The invariant of the algorithm is the fact that prices are set so that in the flow set up, there exists a way to satisfy all demands.

We need to find good starting prices. One idea is

$$p_e = \frac{\min_i m_i}{\sum_j b_j} \forall \text{ good } e$$

It is easy to see that these prices might still not be low enough (some goods may still not be wanted).

So, we set up the graph and if there exists good j that has no edges, lower p_j just until an edge appears. In this way good j has an incoming edge and edges are not removed from the graph.

Claim. The invariant is true at the start.

Proof. Greedy allocation works. (Everything is so cheap, everyone can afford everything). Sell the total quantity of a product to any buyer that wants it.

What goods should *not* raise prices? When increasing all prices by a factor, the ratio $\frac{u^i_j}{p_j}$ stays the same. However, some buyers may run out of money.

We define the *danger set* $S \subseteq$ *goods* such that $\sum_{j \in S} p_j b_j = \sum_{i \in \Gamma(S)} m_i$, where $\Gamma(S) = \{i : \exists j \in S, (i,j) \in E\}$.

Claim. Given j one can find a danger set S containing j via max flow, if one exists.

Algorithm

Repeat until no more danger sets:

1. find a danger set S

2. freeze prices of goods in S
3. remove edges between $\Gamma(S)$ and $\overline{S}$, since buyers in $\Gamma(S)$ are short of money.

No more danger sets $\Rightarrow$ new prices $p_j c$, for $j \notin$ danger set. c is a scalar. (increase prices of products that do not belong to any danger set uniformly)

Raise c until one of the following happens:

- a new edge appears between $\overline{\Gamma}(S)$ and S
- a new node $j \in \overline{S}$ gets into danger set

Need to show:

- new prices satisfy invariant
- there exists a lower bound on the price increase. If all quantities are integers and $u_j^i < U$ then the lower bound would be $\frac{1}{kU^{2k}}$.

CS 684 Algorithmic Game Theory	March 8, 2004
Lecturer: Eric Friedman	*Scribe: Davina Kunvipusilkul*

Solution Concepts and Nash Equilibria

1 Introduction

Consider a game $G = (\mathcal{S}, \pi)$ where $\mathcal{S}$ is the set of strategy and π is the set of payoff.

We are interested in the outcome of the game $\mathcal{O}(G) \subseteq \mathcal{S}$. Note that different set of players can lead to different outcome.

Today we are interested in when $\mathcal{O}(G) = Nash(G)$. We will also consider $max\, Nash(G)$ and $min\, Nash(G)$.

Example: Cournot Duopoly/FIFO Queuing

In this game, users share a network link. Assume the game consists only of a few big players who follow selfish routing.

When $n = 2$, $\mathcal{S} = [0, \infty], \pi_i(\mathcal{S}) = S_i(1 - S_i - S_j)$ and $Nash(G) = \{(\frac{1}{3}, \frac{1}{3})\}$.

This is inefficient since they would be better off at $(\frac{1}{4}, \frac{1}{4})$.

Next, we will consider undominated strategies $\mathcal{U}(G)$. The idea behind this is that some strategies are clearly not good to play since they are dominated.

Example: Prisoner's Dilemma

	C	D
C	1,1	-1,2
D	2,-1	0,0

Since C is strictly dominated by D, $\mathcal{U}(G) = \{(D, D)\}$. This is also Nash.

Note: $Nash(G) \subseteq \mathcal{U}(G)$. That is, we don't want to play stupid strategies.

2 Selection Criteria

(0) Well defined

(1) Obvious outcome

(2) Rationality (therefore predictable)

(3) Learning (Learnable $\rightarrow$ we can get to Nash)
    - distributed algorithm

(4) Myerson Criteria

3 Problems with Nash

(0) **Well defined**

- Nonexistence

  Solution: mixing, convexification

- Multiple equilibria (Strong implementation)

  This one we can't really get rid of. Consider the Schelling example from *Strategy of Conflict*.

  You want to meet a friend in New York. You don't know where to meet. Where should you go?

  There are lots of Nash equilibria, all equally good (and all equally useless), that is when the two people are at the same place.

  Solution Idea: "Refinements." However, most proposed solutions are very specific to the considered problems and generalization does not appear possible.

  What about generating a game matrix at random?

  Then $\mathbb{E}(\#Nash)$ is exponential in everything. For example, if we have 6 players and 6 strategies, then $\mathbb{E}(\#Nash) \approx 5 \times 10^6$ (Mclennan, University of Minnesota).

(1) **Obvious outcome**

"If it is so obvious, why didn't people think about it?"

"Obviousness is to the eye of the beholder."

(2) **Rationality**

Osbourn & Rubenstein: Outcome is *Rationalizable set* (for most games, this is everything). That is, we are in the situation where "I think what I'll do is good for what you'll do which will be good for what I'll do which ..." and this can lead to a very large set of possible outcome.

(3) **Learning**

We want a method that players can follow and converge to Nash.

- No polynomial time algorithm for Nash is known for general problems.
- Fudenberg & Levine, 1993: "Rational learning leads to Nash equilibrium" (with some technical restrictions). However, Nachbar (1997) showed that these technical restrictions are impossible since they require solving doubly exponential problems in every stage.

Examples of Simple Learning

- **Best Reply Dymamics**

  Assume the other players won't change, what should you do? Same for other players.

  Consider the following game:

	l	c	r
t	10,15	0,0	15,10
m	0,0	1,1	0,0
b	15,10	0,0	15,10

  Nash exists in the middle. However, if the players start at top left, this will cycle forever.

  Let $\mathcal{BR}(G)$ be where players end up if the game converges. In the case of FIFO Queuing, the two players will converge.

Aside:

	l	r
t	10,10	10,9
b	9,$-2^1$0	8,8

$\rightarrow Nash = (10, 10)$ but the column player will always play right. So Nash is not Robust

- **Adaptive Dynamics**

  Each player predicts what the other players will play based on history with finite memory. Let $P(h^t)$ be the prediction (must be in the set of previous plays). Each player uses best reply to $P(h^t)$.

  Consider a fictitious play where each player find probablility distribution from past play then gives best reply. This was thought to lead to Nash until Shapley provided a counter example based on Rock/Paper/Scissors.

  Example: Prisoner & Altruist

  Let A be the row player (Altruist) and B be the column player (Prisoner).

	C	D
C	1,1	-1,2
D	0,-1	0,0

  In the first iteration, $\mathcal{U}(G) = \{(C, D), (D, D)\}$.

  Now A knows that B will defect.

  So in the second iteration, A will defect too, so $\mathcal{U}^2(G) = \{(D, D)\}$ is the only rationalizable strategy.

  Theorem 1 *(Miligrom & Roberts, 1992) Let $\mathcal{U}^\infty$ be the serially undominated set. Adaptive learning leads to $\mathcal{U}^\infty$.*

  An extension by Friedman & Shenker (1999) shows that no assumptions are needed for the previous theorem to hold. (Players do not need to know the matrix, etc).

- **Asynchronous Learning**

  Consider the same game as before: (A is the row player and B is the column player)

	C	D
C	1,1	-1,2
D	0,-1	0,0

  What if players play at different speed, will this converge?

  Suppose A is very fast (updating every second) and B is very slow (updating every minute). Also assume that they do not know the payoff matrix.

  Since A can update every second, A can adjust to what B plays. That is,

  If B plays C, A will play C.

  If B plays D, A will play D.

  What happens?

  B will see that when he plays C he gets approximately 1 and when he plays D he gets approximately 0.

  Thus, this will converge to (C, C) – Stackelberg equilibria.

  Generally, this will not converge to Nash.

(4) Myerson Criteria

"What else should I use?"

CS 684 Algorithmic Game Theory
Lecturer: Mark Sandler **March 12, 2004**

1 Graph representation of multiplayer games

Usually n-player games are represented in tabular form. Say if each agent has t actions available, then the game is given by n, n-dimensional matrices, each of size t^n. Matrix i specify the payoff of player i, given the joint actions of all players. Obviously this approach is not very compact, especially if each player payoff depends only on the actions of the small subset of players.

In this lecture we limit ourselves to two-action games, with maximal payoff of 1[1].

Now we describe graph model for multiplayer games. We represent our n-player game as a pair $(G, \mathcal{M})$, where G is unidirected graph with n nodes, where there exist edge between i and j if and only if actions of player i influence actions of player j. The second element $\mathcal{M} = \{\mathcal{M}_1, \dots \mathcal{M}_i\}$, represent payoff matrices for each player. Let $N_G(i) \subseteq \{1, \dots, n\}$ be a set of neighbours of i in G, by definition we include i in $N_G(i)$. Then each $\mathcal{M}_i$ will have dimension $|N_G(i)|$ and hence $2^{|N_G(i)|}$ elements. Say, we bound maximal degree of G by k, then total input size for the game would be order of $O(n2^k)$, the goal is to find NE in time polynomial of the input size. Note that if graph G is a clique, then it represent general case n-player game. So one would expect that problem to be hard for general graphs. Unfortunately even for graph with bounded tree width it is still an open problem, however we can do it for trees.

Even for trees the problem is not trivial, in today's lecture we will present algorithm from the paper [1], which runs polynomial (or quasipolynomial[2]) time and finds all approximate Nash Equilibria (to be defined later) for arbitrary tree. And secondly (if time permits), i will sketch the modification of the algorithm which would allow to find *all* exact equilibria, in exponential time[3]. Natural example for such a game would be a company, where payoff of everyone depends from the person himselves, upper manager actions and action of his subordinates. While actions of CEO influence the payoff of everyone, they do so only by propagating action changes downstream the tree.

2 More Notation

Suppose player i has k neighbors (including himself) and $\vec{x} \in \{0,1\}^k$, $M_i(\vec{x})$ denotes player i payoff when his k neighbors played $\vec{x}$. Analagously for mixed strategy $\vec{p} \in [0,1]^k$, $M_i(\vec{p})$ denotes expected playoff of player i.

For arbitrary vector $\vec{x}$, $\vec{x}[i : p_i]$ denotes vector which is the same as $\vec{x}$, except that i-th coordinate is replaced by p_i.

Definition 1 (Approximate Nash) *We say that set of strategies x defines an approximate nash, with respect to constant ε if no player can benefit by more than ε. In other words for every i and p_i, $\mathcal{M}_i(\vec{x}[i : p_i]) \leq M_i(\vec{x}) + \varepsilon$.*

[1]In general our results would be linearly dependent on maximal payoff

[2]e.g. exp(polylog)

[3]Note, that even this result is not trivial, because no finite algorithm is known for general n-player games.

Since there is correspondence between players and nodes, for simplicity we will also denote by M_V, the payoff matrix of player corresponding to V.

3 Abstract Tree Algorithm

We first present algorithm without specifying data structures (and without even showing that they have finite representation), and then instantiate these structures for approximation algorithm.

First of all since G is a tree, we can root it. Fix some root R. Now every node U (except root) we have uniquely defined parent $P(U)$ and possibly many children. Let G^U denotes a subtree induced by U, and M^U denotes a collection of payoff matrices with players from G^U. If $v \in [0,1]$ denotes some mixed strategy, then let M^U_v denotes the collection of payoff matrices for G^U, except that M_U will be collapsed by one index by fixing the parent of U to play strategy v. By definition we assume that $M^R_v = M^R$.

The idea of the algorithm is to build up a table $t_U(v,u)$ for every node U with the following semantics

$$t_U(v,u) = \begin{cases} 1 \text{ if there is NE in the game } (G^U, \mathcal{M}^U_v)\text{, where } U \text{ plays } u \\ 0 \text{ otherwise} \end{cases}$$

Since mixed strategies form a continuous set, it is not clear how we can represent such a table. However for now we assume that we can do this. Then we can easily build the table starting from the leaves and going up the tree. After we build it we can go back down and reconstruct the Nash Equilibria

Now we present the algorithm to build up tables t_U for all nodes.

Algorithm 1 *Tree Nash.*

```
1 Root the tree.
2 Order nodes starting from the leaves, so that each node is listed only after all its children are
  listed. (Say in reverse depth-first order).
3 For every node $U$ in $t$:
    (a) ($U$ is a leaf.) For every pair of strategies $(v,u)$, set $t_U(v,u)$ to 1 iff $u$ is the best response
        for $U$ if the parent of $U$ plays $v$.
    (b) ($U$ is an internal node.) Let $\{U_1, \dots U_k\}$ be the children of $U$. For all $v,u \in [0,1]$ and
        all joint mixed strategies $\vec{u} = \{u_1, \dots, u_k\}$, we set $t_U(v,u)$ to be 1, iff $t_{U_i}(u,u_i) = 1$ for
        all $i$, and $U = u$ is the best response in the game $(G^U, \mathcal{M}^U_v)$ where children of $U$ are
        playing $\vec{u}$.
```

And to print Nash Equilibrium we just use the following recursive algorithm.

1 Choose a strategy r for root R, such that $t_R(0,r) = 1$.[4]

2 For every node U, with already chosen strategy u, and its parent playing v, we select joint strategy $\vec{u} = \{u_1, \dots u_k\}$ for all its children $\{U_1, \dots U_k\}$ such that:

[4]Note that by definition of $t_R(v,u)$ for root node R, the value of parent strategy v is ignored.

(a) $t_{U_i}(u, u_i) = 1$ for all $1 \leq i \leq k$.

(b) $U = u$ is the best response in $(G^U, \mathcal{M}_v^U)$, where $\{U_1, \ldots U_k\}$ are playing $\{u_1, \ldots, u_k\}$.

By construction of t_U we can alway choose $\vec{u}$.

Note, that by chosing different joint stategies $\vec{u}$ satisfying (a) and (b) we can build all possible NE, and step (3) of the construction algorithm ensures that we can always chose feasible $\vec{u}$ and hence we will have NE at the end.

4 Approximation Algorithm

In this section we use algorithm 1, but we consider only discretized strategies with step τ, and then prove that by taking τ small enough we can achieve ε eqiulibria. (Of course we will have to argue that our algorithm stays quasipolynomial)

We start with the following simple lemma:

Lemma 1 *Let $\vec{p}, \vec{q} \in [0,1]^k$ satisfy $\max_i |p_i - q_i| \leq \tau$, str then provided $\tau \leq 1/k$,*

$$\prod p_i - \prod q_i \leq 2k\tau$$

Proof. Indeed, consider the maximal possible attainable difference between $\prod p_i$ and $\prod q_i$. Without loss of generality we shall assume that $\prod p_i > \prod q_i$, but then for all i, we have $p_i - q_i = \tau$ (otherwise we could have made the difference larger), therefore we have:

$$\prod p_i - \prod q_i = \prod (q_i + \tau) - \prod q_i \leq \sum_i C_k^i \tau^i \leq \sum \frac{k^i \tau^i}{i!} \leq 2k\tau$$

where the first inequality comes from the fact that $q_i \leq 1$, second from upper bound on C_k^i and third from lower bounding $i!$ by 2^i and the fact that $\tau \leq 1/k$. ■

Lemma 2 *Let the mixed strategies $\vec{p}$ and $\vec{q}$ for $(G, \mathcal{M})$ satisfy $|p_i - q_i| \leq \tau$ for all i, then provided that $\tau \leq 1/k$*

$$|M_i(\vec{p}) - M_i(\vec{q})| \leq 2^{k+1} k\tau$$

Proof. Indeed for any $\vec{p}$

$$M_i(\vec{p}) = \sum_{\vec{x} \in \{0,1\}^k} M_i(x) \prod_{i=1}^{k} p^{1-x_i} (1 - p_i)_i^x,$$

therefore:

$$|M_i(\vec{p}) - M_i(\vec{q})| \leq \sum_{\vec{x} \in \{0,1\}^k} |M_i(\vec{x}Z)||\alpha(\vec{x}) - \beta(\vec{x})| \leq \sum_{\vec{x} \in \{0,1\}^k} |\alpha(\vec{x}) - \beta(\vec{x})|$$

where $\alpha(\vec{x}) = \prod_{i=1}^{k} p^{1-x_i} (1 - p_i)_i^x$ and $\beta(\vec{x}) = \prod_{i=1}^{k} q^{1-x_i} (1 - q_i)_i^x$. Since $\tau \leq \frac{1}{k}$, we can immediately apply lemma 1. ■

Lemma 3 *For any Nash Equilibrium p, if $\tau \leq \frac{1}{k}$ then the nearest q in the τ-grid is $2^{k+2}k\tau$-Nash equilibrium for $(G, \mathcal{M})$*

Proof. Indeed, let r_i be the best response for player i to $\vec{q}$. Then we have:

$$M_i(\vec{q}[i : r_i]) <= M_i(\vec{p}[i : r_i]) + 2^{k+1}\tau \leq M_i(\vec{p}) + 2^{k+1}\tau \leq M_i(\vec{q}) + 2^{k+2}\tau$$

where first and last transitions follow from lemma 2 and second from the fact that p is Nash Equilibria. ■

This lemma ensures that if $\tau \leq \frac{\varepsilon}{k2^{k+2}}$, then we will have approximate NE on τ-grid.

Now we describe modifications which should be made to the aglorithm:

- The algorithm now takes additional input ε. We pick $\tau = \frac{\varepsilon}{k2^{k+2}}$
- For every node table $t_U(u, v)$ now contains only τ-discretized strategies.
- Instead of chosing the best strategy we choose ε-best strategy (e.g. which is worse the best strategy by no more than ε)

Lemma 3 assures that there is always exist a ε-nash equilibria, and by picking ε-best response we will always find it.

For running time analysis we note, that each table $t_U(u, v)$ now has size $(\frac{1}{\tau})^2$ and we have to check k whether each element is zero or one. For that we check whether there is an assignment $\{u_i\}$ for the children of U such that u is the best response to the parent playing v and $t_{U_i}(u_i, u)$ is 1. Each check could be done in time $O(2^k)$, so the whole computation could be done in $O(2^k \times \frac{1}{\tau^{k+2}})$. For fixed k, this time is polynomial in $1/\varepsilon$, n and the length of the input. However in general case this time is only quasipolynomial in the length of the input.

5 Exact algorithm

It can be shown that areas where $T_U(u, v)$ is equal to one, can be repsented as a union of at most exponentially (in number of players) many rectangles on the unit square. Moreover it is possible to compute values of $t_U(u, v)$, given $t_{U_i}(u_i, v)$, where U_i are all children of U. This gives all missing steps of the algorithm. We are not going to any further detail here, and refer the reader to [1].

References

[1] M. Keans, M. Littman, S. Singh “Graphical models for Game Theory,” *Proc. International Joint Conference in Artificial Intelligence*, 1999.

CS 684: Algorithmic Game Theory	Friday, March 12, 2004
Instructor: Eva Tardos	
Guest Lecturer: Tom Wexler (wexler at cs dot cornell dot edu)	Scribe: Richard C. Yeh

Network-building

This lecture describes a game that models the building of a network. There are three main points:

1. For the general case of this game, Nash equilibria do not always exist. The finding of Nash equilibria in this game is NP-complete.
2. When Nash equilibria exist in this game, the total cost of building certain Nash-equilibrium networks is between 1 and k (the number of players) times the cost of the optimal network.
3. For a simplified version of this game, the total cost of building the Nash-equilibrium network can be found and is equal to the cost of the optimal network.

For more details, please see E. Anshelevich, A. Dasgupta, É. Tardos, and T. Wexler, “Near-optimal network design with selfish agents”, STOC ’03, available at http://www.cs.cornell.edu/~wexler/.

Introduction

Previously, Professor Tardos presented Roughgarden games, in which players route traffic in a network. The players’ selfish motive: to achieve the shortest routing time.

Today’s lecture will take a different perspective: the building of a network.

The game

We start with a graph representing a network $G=(V, E)$, where we can think of the vertices as servers and the edges as *possible* links. Every edge e has some cost $c_e \geq 0$ needed to install the link, connecting the servers at the ends.

We have k players. Each player i has a pair of nodes s_i, t_i, and is interested only in building just enough of the network to connect those nodes, not in building the entire network.

The player strategies are given by the matrix $p_i(e)$, whose elements represent the statements “player i contributes $p_i(e) \geq 0$ towards the cost of edge e.”

Bought network

First, we’ll check whether edge e is bought. Add up all players’ contributions and check whether this sum is at least the cost of the edge. Define the **bought graph** or **bought network** to be the set B of edges e satisfying: $\sum_i p_i(e) \geq c_e$.

To have the players connect their pairs as cheaply as possible, define each player’s utility u_i to be:

- $-\sum_{e \in B} p_i(e)$ [minus the total amount paid] if he connects his pair

  (regardless of whether all the edges were actually built.);
- $-\infty$ [minus infinity] otherwise (thereby forcing players to connect).

Details and Comments

Today, we just want to connect the pair as cheaply as possible. There is no notion of fairness or capacity.

Today,

- The sources and sinks are not necessarily disjoint.
- The graph is undirected (but the directed case is not much different).
- For all edges, set the edge cost = 1.
- There can be nodes that are not terminals. (We will look at Steiner nodes later in the lecture.)

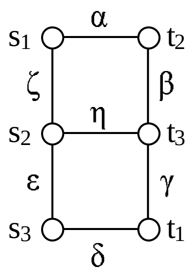
In a real situation, we might have other constraints that we will ignore today, such as:

- Multiple pairs per player
- Need for redundant network links
- Desire for shortest time

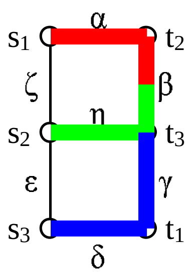
(This will be a full-information game: everyone knows the graph and contributions.)

Plan: Study the Nash Equilibria of this game. What are the stable solutions?

For illustration, consider the following network:

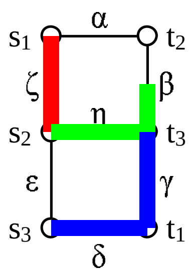


Suppose we arrange the strategies as:



Player	*$p_i(\alpha)$*	*$p_i(\beta)$*	*$p_i(\gamma)$*	*$p_i(\delta)$*	*$p_i(\varepsilon)$*	*$p_i(\zeta)$*	*$p_i(\eta)$*	*u_i*
1	1	0.5	0	0	0	0	0	***–1.5***
2	0	0.5	0	0	0	0	1	***–1.5***
3	0	0	1	1	0	0	0	***–2***
Bought?	Yes	Yes	Yes	Yes	No	No	Yes	

What happens? All players have connected their sources and sinks. However, this is not a Nash equilibrium, because player 1 would prefer to switch his strategy to:



Player	*$p_i(\alpha)$*	*$p_i(\beta)$*	*$p_i(\gamma)$*	*$p_i(\delta)$*	*$p_i(\varepsilon)$*	*$p_i(\zeta)$*	*$p_i(\eta)$*	*u_i*
1	0	0	0	0	0	1	0	***–1***
2	0	0.5	0	0	0	0	1	***–∞***
3	0	0	1	1	0	0	0	***–2***
Bought?	No	No	Yes	Yes	No	Yes	Yes	

Result:

- player 1's utility goes from –1.5 to –1;
- player 2's utility goes from –1.5 to –∞ (fails to connect).

Burning questions

Do Nash equilibria always exist? If so, how expensive are they (compared to an optimum network)? Can we find these equilibria?

All of these questions have disappointing answers.

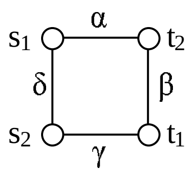
Basic properties of Nash equilibria in this game

Any Nash equilibrium must:

1. buy an acyclic network (a tree or a forest). (If there were a cycle, then players would prefer to buy one fewer edge, not affecting connectivity.)
2. players only contribute to edges on their unique path in the bought network.
3. for any edge e, total payment is either c_e or zero.

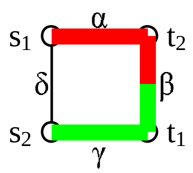
Example with no Nash equilibrium

Consider the following simple example where there is no Nash equilibrium:

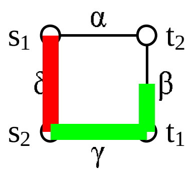


Two players, four nodes (two sinks and two sources), and four edges, each with cost 1. (This is also figure 1 from the paper.)

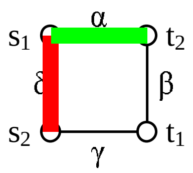
For example, we could begin with both players paying for 1.5 edges:



But then player 1 would prefer to defect:



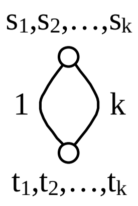
And then player 2 would prefer to change to:



This example shows that Nash equilibria don't necessarily exist.

Example with multiple Nash equilibria

Here's another example: imagine a two-node, two-edge graph, where all k players have the same source and sink nodes. The two parallel links have costs 1 and k, as shown.



There are at least three Nash equilibria:

- Each player could pay 1/k, and the group as a whole buys the cheaper edge.
- Each player could pay 1, and the group as a whole buys the cost-k edge.
- One player could pay 1, buying the cheaper edge, and the other players free-ride.

Calculating the Nash/Opt ratio:

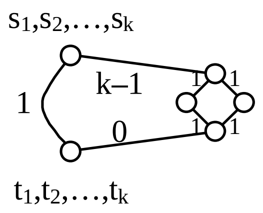
Claim: Any Nash equilibrium costs at most k times the total cost of the optimal solution. (k·cost(OPT)).

Proof: Suppose otherwise — that there exists a Nash equilibrium where the total cost exceeds (k·cost(OPT)). Then there must be at least one player paying more than the optimal total cost. This is a contradiction, because that one player would have preferred to pay just the optimal total cost.

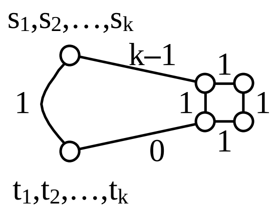
(Recall and compare: in the Roughgarden game, the costs of Nash equilibria were unique, and Nash equilibria always existed.)

Define the **optimistic price of anarchy** to be the ratio of the total cost of the cheapest Nash equilibrium to the total cost of the optimal solution. This quantity indicates how good uncoordinated solutions can be.

Even the best Nash equilibrium can be terrible; for example, combine the two previous examples by inserting the no-equilibrium network into the cost-k edge:



or



(where, as before, the no-equilibrium network players (not included in k) have sources and sinks at opposite corners of the little network) This forces the k players to buy the expensive edge.

Nash claimed that mixed equilibria always exist. In our case, we only consider pure strategies because all players must connect; otherwise, the expected utility is not well-defined. Any given connection is a manifestation of a pure strategy.

Single-source connection game

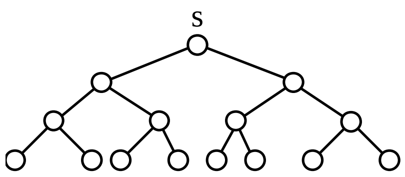
Since finding Nash equilibria is in general NP-complete, we will consider a simpler case:

Define a **single-source connection game** to be one in which all players share the same source node. (For all players *i*, $s_i = s$.) Outside of the game-theoretic context, this is the Steiner tree problem, with the root as the source.

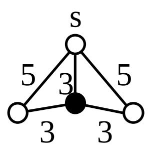
Theorem: in a single-source connection game, Nash equilibria exist, and the optimistic price of anarchy is 1.

Proof sketch:

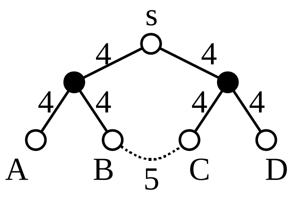
Simple case: all nodes are player-terminals. Imagine starting with a minimum-cost spanning tree. This is stable if every node pays for the edge immediately toward root. If any player were to prefer to buy a different edge (i.e., one not in the minimum-cost spanning tree), then we must not have started with a minimum-cost spanning tree.



Complication: If we add Steiner (non-terminal) nodes (represented in the figures as filled disks), then we must determine a way to pay for the edges from the Steiner nodes toward the root. We must begin with an optimum Steiner tree.



Here, players will pay a maximum of 5.



The payments don't have to be split evenly: players A and D will pay up to 8, while players B and C will only pay up to 5.

The idea is that we can add the payments from the bottom up (from the sinks to the source), while never violating the implicit constraints that a player will pay only as much as her or his cheapest alternative path to the root. This works; below is an argument by contradiction:

What if we ask all the players what they're willing to pay, and it's not enough to buy the optimum Steiner tree? It must be that some player has some cheaper alternate edge to buy than that assigned by the optimum Steiner tree. But if we were to allow this player to deviate, then the total payment for the bought network would be less than for the optimal Steiner tree, which is a contradiction. Either this player or some other player must have lied.

In both cases, the minimum-cost spanning tree or minimum-cost Steiner tree is optimal, and the optimistic price of anarchy is 1.

CS 684 Algorithmic Game Theory	**Scribe: Benjamin Ee**
Instructor: Eva Tardos	**March 15, 2004**

We have the following market equilibrium game: There is a link with limited bandwidth ***B*** (set ***B***=1 w/o loss of generality), and *n* users that want a share of the bandwidth. User *i* has utility $U_i(x_i)$ for x_i units of bandwidth, where $U_i(x_i)$ is a continuous, monotonic increasing and strictly concave function.
Objective: Distribute the bandwidth among all the users so that total demand = ***B****.*

Algorithm 1 *(assumes global knowledge of individual utility functions)*
Generate market clearing price **p** for the bandwidth. If utility is also measured in dollars, then:

- User *i* maximises: $U_i(x_i) - x_i\mathbf{p}$. $U_i(x_i)$ being strictly concave results in a unique max x_i.
- Equilibrium: Set **p** s.t. $\sum_i (x_i) = B$. This is the price s.t when each user decides what to do in isolation, we share everything. It is almost a special case of the exchange economy where the *n+1*th player has 0 utility for everything, and players have no limit in their cash.

Theorem 1 *Let $U_i(x_i)$ be utility function for each of n users demanding bandwidth, total bandwidth be* ***B****, and the unit price of bandwidth be* **p***. Then exists* **p** *such that total bandwidth demanded by users =* ***B****.*

- Let x be the total amount of bandwidth demanded.
- Then we seek to maximize $U_{total}(x) - x\boldsymbol{p}$.
- $$\frac{dUtotal}{dx} = U'_{total}(x) - \mathbf{p} = 0.$$
- Since U'_{total} is continuous, there must exist some value of **p** such that the value of x which sets $\frac{dUtotal}{dx}$ to 0 is ***B***.

Theorem 2 *Equilibrium price* **p** *results in social optimum ($U_{total}(x) - x\mathbf{p}$ is maximized).*

- Consider any allocation $y_1 \dots y_n$.
- Let allocation resulting from equilibrium price **p** be $x_1 \dots x_n$.

$$\sum_i (U_i(y_i) - y_i\mathbf{p}) \le \sum_i (U_i(x_i) - x_i\mathbf{p})$$
$$\therefore \sum_i (U_i(y_i)) - \mathbf{p}B \le \sum_i (U_i(x_i)) - \mathbf{p}B.$$

- Disadvantage of this algorithm is that $U_i(x)$ for all *i* is private information, which users are typically either unaware of, or not willing to disclose.

Algorithm 2

- "*Kelly*" pricing mechanism:
    - *Does not assume knowledge of individual utility functions*
    - Start by obtaining total amount each player is willing to pay for the bandwidth.
    - Let amount each player agrees to pay initially be w_i.
    - Set bandwidth that player i received to $\frac{w_i \times B}{\sum_{all j} w_j}$ (proportional fair sharing).
    - Results in implicit unit price of $\mathbf{p} = \frac{\sum_{all j} w_j}{B}$ for bandwidth.
    - Game:
        - Network gives price $\mathbf{p} = \frac{\sum_{all j} w_j}{B}$ to all users.
        - Users use $\mathbf{p}$ to update their w_i. Assume users are price takers, and do not think about their effect on price. This is realistic if users are assumed to be small enough not to affect prices. Users update prices by attempting to maximize $U_i(\frac{w_i}{B}) - w_i$ (analogous to maximizing $U_i(x_i) - x_i\mathbf{p}$ in first algorithm).
        - Users continue updating prices until Nash equilibrium results where no more updates occur.

Theorem 3 *Equilibrium price **p** resulting from this game will equate supply (**B**) and total bandwidth demanded by all users at **p**.*

- Note that total bandwidth demanded during all iterations of the update loop $= \sum_{all i} (\frac{w_i}{\sum_{all j} w_j}) \times \mathbf{B}$.
  $= \mathbf{B}$.
- Hence, at Nash equilibrium, where users do not want more (or less) bandwidth at the current prevailing prices, then $U_i(\frac{w_i}{B}) - w_i$ is maximized for all users ***at that price***. Therefore, we have the same equilibrium as was reached by the first algorithm.

CS 684: Algorithmic Game Theory	**Scribe: Mikolaj Franaszczuk**
Instructor Eva Tardos	**Wednesday, March 17, 2004**

Bandwidth Sharing

The **bandwidth sharing game** is defined as follows:
There is a single link with capacity B, k users, and a utility function $U_i (x)$ for all users i. U_i must be monotone increasing and concave. For proof purposes, we also make the additional weak assumptions that U_i is strictly concave and differentiable.

There is a pricing mechanism in this game:
Given a price p, each user solves the following problem

$X_i = \max (U_i(x) - px)$

The user is maximizing the difference between the value of the bandwidth he is getting, and the price he is paying for it. Since we assume the utility function is concave, each X_i is unique.

The price p is at equilibrium if $\sum X_i = B$.

Facts (from last lecture):

1. Equilibrium price exists
2. At equilibrium price p, the resulting X_i's maximize $\sum U_i(x_i)$. This is the social welfare optimum.
3. Nash equilibrium of the following game results in equilibrium price.

The Kelly mechanism for playing this game:

User i offers W_i money, so the resulting price is p = $\frac{\sum_i Wi}{B}$

Assumption: Users are "**price takers**" – they ignore their own effect on the price (explained by either the users being dumb, or, mathematically, if there is a large number of them, no individual user has a significant impact on the price).

So the best response in this case is $\mathbf{X_i = \max (U_i(x) - px)}$. So they offer $\mathbf{W_i = X_i * p}$.

Johari & Tsitsiklis Game:

This is the natural version of the game, in which users are "**price anticipators**" – they take the price formula into account when determining their W_i's.

User i offers W_i and gets $\frac{B * Wi}{\sum_j Wj}$ (proportional sharing).

User i happiness is $U_i * \frac{B * Wi}{\sum_j Wj}$ $-W_i$

This game is different, so it doesn't necessarily result in a social welfare optimum.

Question: How does allocation at Nash compare to the social optimum in this game?

Theorem (Johari & Tsitsiklis):

The worst ratio is ¾ (the social value at Nash is no worse than ¾ the value of the social optimum).

What makes a solution a Nash?
Denote price as:

$$p = \frac{\sum_j Wj}{B}$$

$$X_i = \frac{Wi}{p}$$

User problem: Given all W_j, $j \neq i$, maximize $U_i\left(\frac{B * Wi}{Wi + \sum_{j \neq i} Wj}\right) - W_i$

User is at optimum if the derivative is equal to 0, so if we use the chain rule to take the derivative if the above function we get:

$$Ui'\left(\frac{B*Wi}{Wi+\sum_{j\neq i}Wj}\right)\left(\frac{B*Wi}{Wi+\sum_{j\neq i}Wj}\right)'-1=0$$

We note that $\frac{B*Wi}{Wi+\sum_{j\neq i}Wj}=\frac{Wi}{p}=Xi$. Substituting this and taking the derivative of the second term we get

$$Ui'(Xi)*\left(\frac{1}{p}-\frac{B*Wi}{\left(Wi+\sum_{j\neq i}Wj\right)^2}\right)=1$$

Now multiply by the price p on both sides to get:
$Ui'(Xi)'*(1-\frac{Xi}{B})=p$

What is the equilibrium point?
In summary, a Nash allocation is values X_1, X_2 …. X_k and a prize p so that the allocations sum to B, and they satisfy the above equations.

Corollary: A (deterministic) Nash exists.

Since we assumes that Ui' monotone and continuous, the function $Ui'(Xi)'*(1-\frac{Xi}{B})$ is also continuous, and so for any prize p, we get an allocation Xi that satisfies the equation. When p is low the sum of the allocations is less than B. raising p continuously raises the sum, so there is a value at which the sum equals B.

Note that if someone has a small share of the bandwidth, their change in contribution won't affect the price much, but if they have a large share already, additional contribution may depress the price by a lot.

How good is the equilibrium point?

It will come out in the proof of the theorem as follows:

Let the Nash allocation be X_1, X_2 …. X_k
Some people get more than in Opt, and some people get less (since the sum in both is B).

Fact 1: Worst case utility is linear.

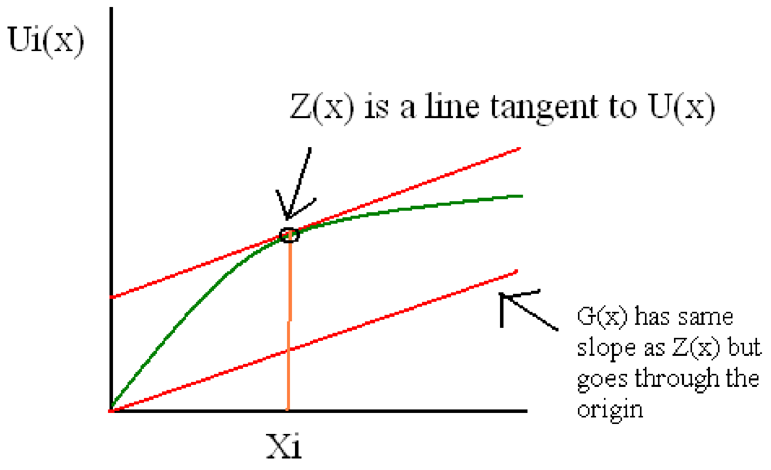
In determining Nash and its value, $U_i'(X_i)$ and $U_i(X_i)$ are the only things that matter. So if we create an alternate utility function Z(x) with:

$$\mathbf{Zi(x) = U_i (X_i) + (x-X_i)*U_i'(X_i),}$$

then we can:
Claim: Utilities Zi have same Nash Xi, and the same value at Nash. The Opt value only improves because Zi(x)>=Ui(x) for all x, as the function Ui is concave by assumption

---

Fact 2: Worst case is when Gi(x) = a_i*x (push the line down to go through the origin)


Ui(x)
Z(x) is a line tangent to U(x)
G(x) has same slope as Z(x) but goes through the origin
Xi

Proof:
Let a new utility function Gi(x) = Ui'(Xi)*x
The Nash is the same place, Xi as before (same reasoning as for fact 1, the only thing that is needed to prove that Xi is a Nash is the value Gi'(Xi) which is the same as Ui'(Xi))

By the same argument the location of Opt in G is the same as the location of opt in Z, as it only depends on the derivative, and pushing the line down doesn't change its slope, so Opt is still in the same location.

What about its values though? A constant gets subtracted from both the Nash and the Opt value. Denote the constant as **C = $\sum_i$Zi(0).** It's subtracted from both Nash and Opt.
The old ratio of Nash to Opt is: $\frac{N}{O}$ and the new ratio is $\frac{N-C}{O-C}$
The new ratio is worse than the original ratio.

If all utilities are linear, the social optimum involved giving the user with the steepest curve all of the bandwidth, and giving nothing to all the other users. So lets say user 1 is the user with the steepest curve, which is:

U$_1$'(x) = $\dfrac{p}{1-\dfrac{X_1}{B}}$

In Nash, we give some amount to other users. The ratio get worse if we make all other user's curves less steep. All other users have Ui'≥p, so the worst case is what all others have **Uj'(x) ≈ p** (and all have very small amounts of bandwidth).

So Opt = $\left(\dfrac{p}{1-\dfrac{X_1}{B}}\right) * B$

Nash = $\left(\dfrac{p}{1-\dfrac{X_1}{B}}\right) * X_1 + (B - X_1) * p$

Hence the ratio Nash/Opt is:

$$\left(\frac{B}{X_1} + \frac{B - X_1}{\dfrac{B}{1-\dfrac{X_1}{B}}}\right) = \left(\frac{\dfrac{1}{(1-a)}}{\dfrac{a}{1-a} + (1-a))}\right) \qquad \text{(by letting a = } \frac{X_1}{B}\text{)}$$

Simplifying we get:

Nash/Opt = $\dfrac{1}{a + (1-a)^2}$

Hence Opt/Nash = $a + (1-a)^2$

What a will cause this ratio to have the worst possible value? Taking the derivative and setting it to 0 we see that 1 – (2*(1-a)) = 0, so a = ½. Plugging that value into the ratio equation, we see the worst ratio is exactly ½ + ¼ = ¾

CS 684 Algorithmic Game Theory	Scribe: Mateo Restrepo
Instructor: Éva Tardos	March 19, 2004

Bandwidth sharing on graphs.

Abstract: We extend the results shown in previous lectures on bandwidth sharing on a single link to the case of multiple links. Two pricing mechanisms are discussed for a simplified version of the game in which users have prespecified fixed paths. The first pricing mechanism is the natural extension of Kelly's protocol in which users are price takers. We show that an equilibrium allocation in this mechanism constitutes a social optimum. The second mechanism was proposed recently by Johari and Tsitsiklis. This defines a competitive game between k players, the users that no longer act as price takers. We state without full proof the existence of Nash equilibria for this game as well as $\frac{3}{4}$ bound on the ratio between the social benefit of a Nash equilibrium and that of the social optimum.

1 Bandwidth sharing on graphs — problem formulation.

In the bandwidth sharing problem on graphs we are given a graph $G = (V, E)$ with a bandwidth (capacity) b_e for each edge. There are k users $i = 1, \dots, k$, each having a source-sink pair s_i-t_i, and a utility function $u_i(x)$ that measures his benefit from a bandwidth x (between s_i and t_i). For the main results that we will develop we will assume that $U_i(x)$ are strictly increasing and strictly concave functions. To keep the mathematics simple we will further assume that these functions are strictly concave and differentiable.

Having said this we present two different versions of the bandwidth sharing problem, the first one simpler than the other.

1.1 Version 1

In this version we are further given fixed paths P_i between s_i-t_i along which each user must route all his traffic. The goal of the problem is to allocate bandwidths $x_1, \dots, x_k$ to the users in order to maximize the total social welfare without violating capacity constraints on the edges of the graph. The bandwidth sharing problem can be written formally as the following (convex) optimization problem.

$$\begin{array}{rcl} \max\limits_{\vec{x}} & & \sum\limits_i U_i(x_i) \\ \text{s. t.} & & \\ x_i & \geq & 0, \forall i \\ \sum\limits_{i: e \in P_i} x_i & \leq & b_e, \forall e \in E \end{array}$$

1.2 Version 2

In this version there are no predefined paths for the users and, furthermore, the users are allowed to split their traffic between different paths.

This version of the problem is harder to deal with, and therefore in what follows, we will focus on the first version. However, all results extend to this version as well.

Yet another version, we would not allow users to split traffic between different paths. This version is significantly harder, and the results do not extend to this version.

2 Kelly's game

The problem defined above is a global optimization problem. In what follows we are going to define games that "distribute" the problem among the users and allow them to find optimal or close to optimal solutions based on "local" calculations.

Kelly suggested the following pricing mechanism. Every user offers a certain amount of money w_i^e to each edge e on his path. The edge e then considers all offers and splits bandwidths proportionally. Given the offers of all users to edge e, the bandwidth that i gets is given by

$$x_i^e = \frac{w_i^e}{\sum_j w_j^e} b_e,$$

and is therefore different for each e. The actual bandwidth that user i will be able to use is just $\min_{e\in P_i} x_i^e$ and his benefit is

$$U_i\left(\min_{e\in P_i} x_i^e\right) - \sum_e w_i^e.$$

This game as such is still not well defined. What happens if an edge doesn't receive any monetary offers, that is, if w_j^e? This might happen, for instance, when there is an edge that only appears in one of the users' paths, say in P_1. In this case, no matter how small the offer of player 1 on these edge is, he will get the totality of the bandwidth there. Therefore, the user will always consider that he is wasting the amount of money offered to this edge.

We fix this bug in the mechanism by declaring that on a non-competed ($\sum_j w_j^e = 0$) edge, on which user i claims an amount x_i^e of the bandwidth, we allocate this bandwidth to each of them, only if we have $\sum_j x_j^e \leq b_e$ (otherwise we allocate 0 to all of them). Intuitively, if $\sum_j x_j^e > b_e$ then the edge presents a bottleneck and it better be the case that some users make some offers.

In this setting **Kelly's game** assumes that users are *price takers*. That is, the users don't anticipate the effect of their payment on the prices, but accept the price on every edge as a given fixed quantity p_e. Based on this price each user evaluates the cost of the bandwidth for him and solves the following (one-dimensional) problem to find the bandwidth that gives him the best payoff

$$\max_{x\geq 0}\left(U_i(x) - x\sum_{e\in P_i} p_e\right)$$

Denoting by x_i the optimal bandwidth that solves this problem, the user offers $w_i^e = p_e x_i$ for edge e, and the router sets $x_i^e = x_i$ if $p_e = 0$.

A "Nash" equilibrium[1] in this game is a choice of prices $\vec{p}$ together with allocated bandwidths $\vec{x}$ such that the prices p_e determine bandwidths x_i as just described (the x_i maximize the users' payoffs), while the offers $w_i^e = p_e x_i$ generate prices p_e again, according to the formula $p_e = \frac{1}{b_e}\sum_i w_i^e$.

We state without proof

Lemma 1 *An equilibrium exists for this game (and is unique if the functions are strictly concave)*

Although we don't prove it we give some ideas for the proof. The first idea is to realize that the prices p_e can be seen as dual variables for the inequalities $\sum_{i:e\in P_i} x_i \le b_e$ and then apply results from duality theory (Lagrange multipliers).

Another approach for the proof would be to use the machinery of Brower's fixed point theorem as it's traditional in game theory.

Easier to prove is the following

Lemma 2 *The resulting allocation at equilibrium is socially optimal.*

Proof. Let $(x_1,\ldots,x_k)$ be the equilibrium allocation and $(x_1^*,\ldots,x_k^*)$ be any other allocation. By definition, the equilibrium allocation satisfies, for every i, $U_i(x_i^*) - x_i^*\sum p_e \le U_i(x_i) - x_i\sum p_e$, (where the p_e are the prices corresponding to the offers $w_i^e = x_i p_e$ in the equilibrium allocation). Adding up these inequalities over all the users we get

$$\sum_{i=1}^{k}\left(U_i(x_i^*) - x_i^*\sum_{e\in P_i} p_e\right) \le \sum_{i=1}^{k}\left(U_i(x_i) - x_i\sum_{e\in P_i} p_e\right) \tag{1}$$

For the Nash bandwidths we have in the second term $\sum_i x_i \sum_{e\in P_i} p_e = \sum_e p_e \sum_{i:e\in P_i} x_i = \sum_e p_e b_e$ since, for every e with $p_e \neq 0$ we have $\sum_i x_i = b_e$. On the other hand, it is always the case that $\sum_{i:e\in P_i} x_i^* \le p_e$ for an arbitrary assignment. This gives us $\sum_{i=1}^{k} x_i^* \sum_{e\in P_i} p_e = \sum_{e\in E}\sum_{i:e\in P_i} x_i^* \le \sum_e p_e b_e$. Combining the last two results we conclude that

$$\sum_{i=1}^{k} x_i^* \sum_{e\in P_i} p_e \le \sum_{i=1}^{k} x_i \sum_{e\in P_i} p_e,$$

which together with (1) gives us

$$\sum_{i=1}^{k} U_i(x_i^*) \le \sum_{i=1}^{k} U_i(x_i)$$

Thus the optimality of the equilibrium assignment. ∎

3 Implementing the Mechanism

Finally, let us comment on implementing Kelly's mechanism. We note that it can be implemented in a simple distributed manner. So the users need to know the total prize $\sum_{e\in P_i} p_e$ to be able to solve the user optimization problem. So the routers on the edges have to somehow send this information along the paths. An edge e need to know the offers w_i^e for all users i on the edge to be able to determine the prize. This is a different number for each edge along P_i, however, the

[1] more properly, "a competitive equilibrium"

users can transmit this information by simply transmitting the chosen rate x_i (where w_i^e is then $x_i p_e$). One can even interpret some of the existing router mechanisms as implementing such prizing games. The idea is that at a reasonable approximation packet loss rate along a paths is the sum of the loss rate along edges. So we can think of the end to end loss rate as encoding the sum of the edge prizes. For the user problem, we can think of different back-off protocols as solving the corresponding user problem with different user utilities. See the paper [3] for more on this.

4 Johari and Tsitsiklis game

As in the previous lecture we go on now to consider the more complicated model that results when we relax the assumption that users are price takers. Johari and Tsitsiklis proposed this game, in which users anticipate the result that their offers will have on the price, as a better approximation of the real problem. Again we are interested in the existence of Nash equilibria for this game, and in the ratio between the optimal social cost and the cost of a Nash equilibrium.

The formal model is as follows. Every user has complete control on his own bids on edges and also realize that the price on each edge is calculated according to $p_e = \frac{1}{b_e}\sum_i w_i^e$. Given the bids $w_{\bar{i}}^e$ of all other users, user i determines his bids w_i^e, on all edges $e \in P_i$, as those maximizing his benefit. Using the notation of [2] (w_i denotes the vector of values w_i^e, $e \in P_i$, $\mathbf{w}_{\bar{i}}$ denotes the set of bids of players other than i, and $Q_i(w_i, \mathbf{w}_{\bar{i}})$ stands for i's benefit, as defined below), the problem solved by each player is

$$\begin{aligned}
\max \qquad & Q_i(w_i, \mathbf{w}_{\bar{i}}) \equiv U_i\left(\min_{e\in P_i} \frac{w_i^e}{w_i^e + \sum_{j\neq i} w_j^e} b_e\right) - \sum_{e\in P_i} w_i^e \\
w_i^e \quad & \geq \quad 0, \forall e \in P_i
\end{aligned}$$

Notice that this is a multivariate (convex) optimization problem.

The latter defines a game between k players. A Nash equilibrium for this game is a set of bids w_i^e that maximizes the benefit of each player under the assumption that the other players won't change their bids. Formally, w_i^e constitute a Nash equilibrium if for every player i

$$Q_i(w_i, \mathbf{w}_{\bar{i}}) \geq (w_i', \mathbf{w}_{\bar{i}}), \quad \text{for all } w_i' \geq 0$$

Again we have

Lemma 3 *A Nash equilibrium exists.*

We will not prove this result but just comment that the existence follows from the application of standard fixed point procedures to a family of "perturbed" games parameterized by ε that converges in a certain sense to our game. After the a limiting procedure a sequence of Nash equilibria for these games converges to a Nash equilibrium for the present game. For more details we refer the interested reader to Theorem 6. p. 20 of [2].

Finally, we have the following result about the price of anarchy in this situation.

Theorem 4 *Let x_i be a Nash allocation of bandwidths to users. Then the social benefit derived from is at least $\frac{3}{4}$ the maximum possible social benefit. In symbols*

$$\sum_i U_i(x_i) \geq \frac{3}{4} \max_{x_i^*} \sum_i U_i(x_i^*),$$

were the maximum on the right hand side is taking over all assignments that satisfy bandwidth limits.

Proof sketch. We need to prove that there is a Nash equilibrium. For now we concentrate on comparing an equilibrium to the optimal solution.

In a Nash equilibrium, the bandwidth that a user gets in different links across his path must be the same. For if there were any edge giving a greater bandwidth to this user, then the user could lower his bid, still retaining the value of $U(x_i)$ but increasing his benefit. From this we conclude that for the purposes of finding a Nash equilibrium, the maximization problem solved by each user is not actually a multivariate problem. There is only one variable that captures all what is essential to user i and that is the (uniform) bandwidth x_i assigned to his path.

Now consider the optimization faced by user i, given that all other users bids are fixed. Each edge has an amount W^e offered to the edge by other users. In order to get bandwidth x this user has to offer payment w_i^e such that $x/b^e = w_i^e/(W^e + w_i^e)$, so user will pay $w_i^e = W^e x/(b_e - x)$ for an edge e along his path. Now his overall utility will be

$$U_i(x) - \sum_{e \in P_i} w_i^e = U_i(x) - \sum_{e \in P_i} W^e x/(b_e - x).$$

By our assumption that the utility functions are convex, this gets maximized when we set the derivative to 0. So in a Nash equilibrium, we must have that user i's bandwidth x_i satisfies.

$$U_i'(x_i) - \sum_{e \in P_i} (W^e/(b_e - x_i) + W^e x/(b_e - x_i)^2) = 0.$$

As we have done in the case of a single edge, we simplify this expression by substituting, the edges unit price: $p_e = (W^e + w_i^e)/b_e = W^e/(b_e - x_i)$ to get:

$$0 = U_i'(x_i) - \sum_{e \in P_i} (p_e + p_e x_i/(b_e - x_i)) = U_i'(x_i) - \sum_{e \in P_i} \frac{p_e}{1 - x_i/b_e}.$$

Recall from the previous lecture that, in the case of a single link of capacity B, an assignment $(x_1, \ldots, x_k)$ is in equilibrium if, for each i, $U_i'(x_i)\left(1 - \frac{x_i}{B}\right) = p$ where $p = \frac{B}{\sum w_i}$ was the price per unit of capacity on the link. Our calculation above shows that for the present case we get analogously that the assignment of bandwidths is in equilibrium if and only if, for each user i,

$$U_i'(x_i) = \sum_{e \in P_i} \frac{p_e}{1 - x_i/b_e} \tag{2}$$

In last expression, the role of each one of the edges is nicely separated in each of the terms of the sum. This now allows us to proceed in more or less the same way as in the case of a single link and obtain the desired result ⊓

5 Complementary References

Most of this lecture was based on the first reference which extended the results of the second reference.

1. F. P. Kelly, *Charging and rate control for elastic traffic,* European Transactions on Telecommunications, vol. 8, pp. 33-37, 1997.

2. Ramesh Johari, and John N. Tsitsiklis. (2004). *Efficiency loss in a network resource allocation game.*

3. S. H. Low, Larry Peterson, and Limin Wang: Understanding Vegas: A Duality Model, Journal of the ACM, 49 (2) 2007-235, 2002.

CS 684 Algorithmic Game Theory	**Scribe: Sumit Kunnumkal**
Instructor: Eva Tardos	**Monday, March 29, 2004**

1 Coalitional Games

In today's lecture, we will look at coalitional games. In a coalitional model, we focus on what groups of players can achieve rather than on what individual players can do. A stability requirement for a coalitional game is that the outcome be immune to deviations by groups of players, i.e., no subset of players can unilaterally improve their outcome. A strong criterion is that all players in the group are strictly better off. A weaker criterion is that the outcome improves for at least one player in the group and makes it no worse for all others. Some applications of coalitional games are:

- Cost sharing for network design: Users benefit from being connected to a server. So they have to build up a broadcast tree. However, it costs to maintain the server/network and the question is how to share the costs.
- Queue management: Multiple users want to route traffic through a switch, which has a flow dependent delay (cost). The queueing delay cost has to be shared among the users. **This is similar to the bandwidth sharing game which we looked at in an earlier lecture, except that the utility need not be a separable function and the cost sharing need not be proportional**.

We now consider a simple version of a coalitional game, namely a coalitional game with transferable payoff **(transferable payoff means that there is no restriction on how the total payoff may be divided among the members of a group)**.

1.1 Coalitional games with transferable payoff

We have a finite set N of users and a function $v : 2^N \to \mathcal{R}^{\geq 0}$. So, v assigns a non-negative number to every $S \subseteq N$. We can think of $v(S)$ as the total payoff available to the members of set S. A natural question that arises is how to share the value $v(N)$ among all the users so that there is no incentive to deviate. Before looking at rules for stable sharing, we consider some examples.

- An expedition of n people discover treasure. It requires two people to carry out one piece of the treasure, in which case the value of the treasure is equally shared between the two. For a subset S, the value is given by
$$v(S) = \lfloor \frac{|S|}{2} \rfloor$$
Let's see if there is way of sharing the value in a stable manner. If $|N| = 2$, then clearly (1/2, 1/2) is a stable sharing. What happens when $|N| = 3$? Note that (1/2, 1/2, 0) is not a stable sharing since the third person can offer, say the second person, a little more than half. Similarly (1/3, 1/3, 1/3) is not a stable sharing since two players can each get a value of 1/2 if they form a coalition. It can be shown there is no stable sharing now.

- Majority vote: A subset S gets a value of 1 if it consists of a majority of the players and nothing otherwise. So $v(S)$ is given by

  $$\begin{aligned} v(S) &= 1, \text{ if } |S| \geq |N|/2 \\ &= 0, \text{ otherwise} \end{aligned}$$

  It can be shown that there is no stable sharing when $|N| \geq 3$.

1.1.1 Rules for stable sharing

The Core: The core is a solution concept for coalitional games, analogous to Nash equilibria for non-cooperative games. For a coalitional game with transferable payoff, a cost sharing is in the core if no coalition can obtain a payoff which is better than the sum of the members' current payoffs.

We note that a cost sharing x has $x_i \geq 0\ \forall i \in N$ and $\sum_{i \in N} x_i = v(N)$. Then, the condition for a cost sharing x to be in the core is that $\forall S \subseteq N,\ \sum_{i \in S} x_i \geq v(S)$. Equivalently, there is no set S and payoff vector y with $\sum_{i \in S} y_i = v(S)$ for which $y_i > x_i\ \forall i \in S$.

Frequently, the core is empty (as in the two examples that we considered). So we want to determine when the core is not empty. Observing that the core is characterized by a set of linear inequalities, we have the following result, referred to as the Bondareva-Shapley theorem.

Theorem 1 *A coalitional game with transferable payoff has a non-empty core iff* $\forall\ y_S \geq 0$ *if* $\sum_{S: i \in S} y_S = 1\ \forall i$, *then* $\sum_S y_S v(S) \leq v(N)$

Proof.
We use the following fact about linear programs:

$$\exists x, x \geq 0, Ax \leq b \text{ iff } \forall y \geq 0, \text{ if } y^T A = 0, \text{ then } y^T b \geq 0$$

Here A is an mxn matrix and x, b, y are nx1, mx1, mx1 vectors, respectively.
For a cost sharing x to be in the core, we need

$$x_i \geq 0\ \forall i \in N, \sum_{i \in N} x_i = v(N) \text{ and } \sum_{i \in S} x_i \geq v(S)\ \forall S \subseteq N$$

Noting that $\sum_{i \in N} x_i = v(N)$ can be written as $\sum_{i \in N} x_i \leq v(N)$ and $\sum_{i \in N} x_i \geq v(N)$ and that the latter inequality can be omitted, the problem reduces to finding x with

$$x_i \geq 0\ \forall i \in N, \sum_{i \in N} x_i \leq v(N),\ -\sum_{i \in S} x_i \leq -v(S)\ \forall S \subset N$$

Applying the above fact about linear programs, we get that the core is non-empty iff $\forall\ y_S \geq 0$, if $\sum_{S: i \in S} y_S = y_N\ \forall i$, then $\sum_S y_S v(S) \leq y_N v(N)$. **Dividing the inequalities by y_N, we get the required result.**

■

We provide some intuition for the theorem. Note that for the core to be non-empty, it must be the case that for any partition of N, say $A_1, \ldots, A_k$, $\sum v(A_i) \leq v(N)$. Otherwise, there is an incentive for a subset of users to break away. Now, consider the case when $y_S \in \{0, 1\}$. Then $\sum_{S: i \in S} y_S = 1\ \forall i \in N$ implies that subsets S with $y_S = 1$ form a partition of N. So $\sum_S y_S v(S) \leq$

$v(N)$ is precisely the partition inequality. **When y's are fractional, the meaning is less clear, but we can think of $\sum_S y_S v(S) \leq v(N)$ requiring the expected value of the "partition" not to exceed v(N).**

In the next lecture, we will look at solution concepts for cost sharing when the core is empty.

CS 684 Algorithmic Game Theory	**Scribe: Frans Schalekamp**
Instructor: Eva Tardos	**March 31, 2004**

Cost sharing — Shapley Value

Last time we looked at sharing value in such a way, that no subgroup of players would get more value if they would just be by themselves. Value (or cost) sharing with this property is said to be **in the core**.

The main problem with the core as we remarked last time, is that it is empty in almost all interesting cases. Today we will introduce another way of sharing cost, called the **Shapley value**, and relate this concept to the core.

1 Setup

Let N be a set of people. Let $c : 2^N \to \mathbb{R}^{\geq 0}$ be a (set)function defining the cost for each subset of N. c is assumed to be monotone (i.e. $c(S) \geq c(S')$ for all $S' \subseteq S \subseteq N$), and $c(\emptyset) = 0$.

The **goal** is to find nonnegative **cost shares** $\{x_i\}_{i \in N}$ such that

$$\sum_{i \in N} x_i = c(N) \qquad \text{(budget balance)}$$

2 Last lecture

Last time, our additional restrictions were

$$\sum_{i \in S} x_i \leq c(S) \qquad \text{for all } S \subseteq N.$$

$\{x_i\}_{i \in N}$ satisfying this is **in the core**.

3 Axiomatic deduction of the Shapley Value

Definition The *marginal cost* for i with respect to $S \subset N$, $i \notin S$, is defined as

$$\Delta_i(S) = c(S \cup \{i\}) - c(S).$$

Consider the following axioms, that we (may) want our cost shares to satisfy.

Axiom 1 ("Dummy axiom") If i is such that $\Delta_i(S) = c(\{i\})$ for all $S \not\ni i$, then

$$x_i = c(\{i\}).$$

Axiom 2 ("Symmetry") If $i \neq j$ such that $\Delta_i(S) = \Delta_j(S)$ for all $S \not\ni i, j$, then

$$x_i = x_j.$$

Axiom 3 ("Linearity") Suppose $c(S) = c_1(S) + c_2(S)$ where c_1 and c_2 are also nonnegative monotone setfunctions with $c_1(\emptyset) = c_2(\emptyset) = 0$, and $\{x_i^1\}_i$ are cost shares for c_1-cost and $\{x_i^2\}_i$ are cost shares for c_2-cost, then

$$x_i = x_i^1 + x_i^2$$

for all i, defines the cost shares for the c-cost.

4 Discussion of the axioms

Axiom 1 says something like if the cost of i is orthogonal (independent) of any other costs, then the cost share of i should be exactly this cost. This seems intuitively reasonable, and is actually implied in the core inequalities (namely for $S = \{i\}$ and $S = N\backslash\{i\}$, combined with budget balance).

Axiom 2 is enforces some kind of fairness: if two players are (truly and utterly) indistinguishable, they should be charged the same. Note the "truly and utterly": there are actually stronger fairness results, which we will do in class Friday.

Finally, the meaning of axiom 3 is kind of unclear, especially in the case of value, instead of costs. In the case of costs, we can think of orthogonal services (e.g. internet connection and cable television) for both of which the players have to pay to the same company. But still it is unclear (or even questionable) that we would want our cost shares to satisfy this property. It does have one big advantage: it gives rise to the following theorem.

5 Shapley Value

Theorem 1 *There is a unique way to satisfy axioms 1, 2 and 3 (and budget balance), called the* **Shapley Value**.

In defining the Shapley Value, we will use the notion of **ordered marginal costs**.

Definition Given an "arrival order" (permutation) σ of the elements in N, let

$$\begin{aligned} x_1^\sigma &= c(\{\sigma_1\}) \\ x_i^\sigma &= c(\{\sigma_1, \sigma_2, \ldots, \sigma_i\}) - c(\{\sigma_1, \sigma_2, \ldots, \sigma_{i-1}\}) \\ &= c(\{\sigma_1, \sigma_2, \ldots, \sigma_i\}) - \sum_{\ell=1}^{i-1} x_\ell^\sigma \end{aligned}$$

for all i. $\{x_i^\sigma\}_i$ is said to be the **ordered marginal costs**.

Properties of ordered marginal costs:

- $x_i^\sigma \geq 0$ (since c increasing)
- $\sum_i x_i^\sigma = c(N)$ (by definition of ordering marginal costs)
- ordered marginal costs satisfy axioms 1 and 3 (provided that c_1 and c_2 use the same order)
- ordered marginal costs are not fair (think of buy-at-bulk: the players that arrive early have to pay huge setup costs, or delay on a link: the players that arrive late will have long delays).

Definition The **Shapley Value** is defined as

$$\mathbb{E}(x_i^\sigma)$$

where the expectation is taken over σ, the probability distribution of which is defined to be the uniform probability distribution of all possible permutations of N.

The following claim follows from the properties of ordered marginal costs, and the fact that σ is chosen uniformly at random.

Claim The Shapley Value satisfies axioms 1, 2 and 3.

This is half of our theorem. The other half is our next claim:

Claim No other cost sharing satisfies axioms 1, 2 and 3.
Proof Consider the following cost functions:

$$c_T(S) = \begin{cases} 0 & \text{if } S \supseteq T \\ 1 & \text{otherwise.} \end{cases}$$

The only way to satisfy axioms 1 and 2 is

$$x_i^T = \begin{cases} 0 & \text{if } i \notin T \quad \text{(by axiom 1)} \\ 1/|T| & \text{if } i \in T \quad \text{(by axiom 2).} \end{cases}$$

The next claim will finish the proof:

Claim All monotone costs c can be written uniquely as $\sum_T \lambda_T c_T$.
Proof Note that c is a $2^{|N|} - 1$ dimensional vector, and we have exactly that many c_T functions. So our claim boils down to showing that the c_T's are linearly independent.

Suppose, by contradiction, that the c_T's are not linearly independent, i.e. there exist nontrivial $\{\lambda_T\}_{\emptyset \neq T \subseteq N}$ such that $\sum_T \lambda_T c_T = 0$. Let S be a smallest set such that $\lambda_S \neq 0$. Note that

$$\sum_T \lambda_T c_T(S) = \lambda_S, \text{ since}$$

$$\begin{aligned} c_T(S) &= 0 \qquad \text{when } S \not\supseteq T, \text{ and} \\ \lambda_T &= 0 \qquad \text{when } S \supseteq T, \end{aligned}$$

contradicting the fact that every element of $\sum_T \lambda_T c_T$ equals 0. ∎

6 Last remarks

First of all, the Shapley Value is in some sense orthogonal to the core. The core maybe empty, the Shapley Value always uniquely exists. In the case of a nonempty core, the Shapley Value may actually lie outside of the core altogether.

Note that the Shapley Value may be unfair: think, for example, about the case when a link is shared between two jobs, one of which is small, the other big. Since the arrival order is uniformly at random, the small job is punished for the big job half of the time, even though it has nothing to do with this big job.

Next lecture we will introduce other (stronger) fairness assumptions.

CS 684 Algorithmic Game Theory	Scribe: David Martin
Instructor: Eva Tardos	Friday, April 2nd, 2004

1. Fair Cost Sharing

Previously, we looked at core cost-sharing and Shapley value cost-sharing.

In the case of core cost-sharing, we have the following theorem.

Theorem. *If the cost function is concave, then the core is non-empty.*

Today, we will discuss *serial cost-sharing* on *convex cost functions.*

1.1 Setup

There are k users $1, \ldots, k$.

User i selects a quantity q_i, for $i = 1, \ldots, k$.

The total quantity is $q = \sum_{i=1}^{k} q_i$.

A non-negative, increasing cost function $C(q)$ returns the cost for producing that quantity.

We will assume that the cost function is convex, such as the queuing delay cost-function:

$$C(q) = 1/(u - q)$$

where u is the capacity of the link that is being shared between users.

User i is made to pay ς_i, which is a share of the total cost.

Note that ς_i is a function of $q_1, \ldots, q_k$.

The utility of user i is denoted by $U_i(q_i, \varsigma_i)$.

We assumes that U_i increases as q_i increases, and decreases as ς_i increases.

1.2 Cost-sharing schemes

The following are some ways of sharing cost.

Average cost

$$\varsigma_i = (q_i/q)C(q)$$

Average cost cost-sharing is immune to users splitting into many tiny users.

But it is not very fair to tiny users who arrive early if C is very convex.

Marginal cost

$$\varsigma_i = C(q)/k + (q_i - q/k)C'(q)$$

(Note that we assume that C is differentiable.)

Marginal cost cost-sharing is immune to users repartitioning their demand.

In general, cost shares with this property are of the form:

$$\varsigma_i = C(q)/k + (q_i - q/k)H(q)$$

for some function $H(q)$.

Serial cost [Moulin-Shankar]

Assume $q_1 \leq q_2 \leq \cdots \leq q_k$.

The cost-shares are defined inductively as:

$$\begin{aligned} \varsigma_1 &= \frac{C(kq_i)}{k} \\ \varsigma_{i+1} &= \frac{C(q_1 + \cdots + q_i + (k-i)q_{i+1}) - (\varsigma_1 + \cdots + \varsigma_i)}{k-i} \end{aligned}$$

Claim. $\sum_{i=1}^{k} \varsigma_i = C(q)$ *and, for each* $i = 1, \ldots, k$, *we have* $\varsigma_i \geq 0$.

Proof.

Directly from the definition of ς_k, we have:

$$\varsigma_k = C(q_1 + \cdots + q_k) - (\varsigma_1 + \cdots + \varsigma_{k-1})$$

Hence $C(q_1 + \cdots + q_k) = \varsigma_1 + \cdots + \varsigma_k$.

All that remains is to show that each $\varsigma_i \geq 0$.

By definition of ς_i, it is enough to show, for each i, that

$$\varsigma_1 + \cdots + \varsigma_{i-1} \leq C(q_1 + \cdots + q_{i-1} + (k-i+1)q_i)$$

This follows by straightforward induction on i. (Note that the base case holds directly from the definition of ς_1 and from the fact that C is a non-negative function.) ■

1.3 Nash games under serial cost-sharing

The user problem for user i is to find q_i, a solution to:

$$\max_{q_i}(U_i(q_i, \varsigma_i(q_1, \ldots, q_k)))$$

where each q_j is fixed for $j \neq i$.

For convenience, we assume that U_i is differentiable and strictly concave. (This ensures that each user problem to has a unique maximum.)

The following is an algorithm to find deterministic Nash.

Algorithm.

Step 1.

Make each user assume that that all users will select the same quantity.

So the modified user problem for user i is to find q_i, a solution:

$$\max_{q_i}(U_i(q_i, \varsigma_i(\underbrace{q_i, \ldots, q_i}_{k}))$$

Renumber the users such that q_1 is the least quantity selected by any user under the modified user problem.

Step $i+1$.

Fix the quantities selected by users $1, \ldots, i$ to be $q_1, \ldots, q_i$ as calculated in previous steps.

Make each user $j > i$ assume that all users $j' > i$ will select the same quantity.

So the modified user problem for user $j \geq i$ is to find q_j, a solution:

$$\max_{q_j}(U_i(q_j, \varsigma_i(q_1, \ldots, q_i, \underbrace{q_j, \ldots, q_j}_{k-i}))$$

Renumber users $j > i$ such that q_{i+1} is the least quantity selected by any user $j > i$ under the modified user problem.

Theorem. *The solution $q_1, \ldots, q_k$ produced by the above algorithm is a Nash.*

Proof.

If $q_1 \leq \cdots \leq q_k$, then a simple backwards induction on i shows that each user sets the optimal value for themselves. So all that remains is the following claim.

Claim. *The solution $q_1, \ldots, q_k$ produced by the algorithm are such that $q_1 \leq \cdots \leq q_k$.*

Proof.

Suppose not.

Let i be minimal such that $q_i > q_{i+1}$.

So $q_1 \leq \cdots \leq q_i$, since i is minimal.

Note that, as total quantity increases, ς_{i+1} increases, and hence U_{i+1} decreases. Hence, since $q_i > q_{i+1}$, we have:

$$U_{i+1}(q_{i+1}, \varsigma_{i+1}(q_1, \ldots, q_{i-1}, q_i, \underbrace{q_{i+1}, \ldots, q_{i+1}}_{k-i})) \leq U_{i+1}(q_{i+1}, \varsigma_{i+1}(q_1, \ldots, q_{i-1}, \underbrace{q_{i+1}, \ldots, q_{i+1}}_{k-i+1}))$$

Hence, for user $i+1$, the maximal utility at step i was no less than the maximal utility at step $i+1$.

Hence, for user $i+1$, the optimal quantity at step i was no more than the optimal quantity at step $i+1$ (namely q_{i+1}), since, by assumption, U_{i+1} is strictly concave.

But, since $q_{i+1} < q_i$, this contradicts the minimality of q_i at step i in the algorithm. ■

Remark. *The solution $q_1, \ldots, q_k$ produced by the above algorithm is, in fact, the* ***only*** *Nash, even over all mixed strategies.*

CS 684 Algorithmic Game Theory	**Scribe: Alice Cheng**
Instructor: Eva Tardos	**April 5, 2004**

1 Serial Cost Sharing

Recall from the previous lecture the serial cost-sharing setup: We have players, 1, ..., k, who decide on quantities $q_i \geq 0$. There is an associated cost function C, and total cost $C(q)$, where $q = \sum_{i=1}^{k} q_i$. We will assume that C is strictly convex, differentiable and monotone increasing. For each player, we have cost shares ς_i such that $\sum_{i=1}^{k} \varsigma_i = C(q)$. We also have player utilities, u_i. In the last lecture, we used a more general form for the utility functions: For each player i, we have $u_i(q_i, \varsigma_i)$ where u_i is strictly concave, differentiable, monotone increasing with respect to q_i and monotone decreasing with respect to ς_i.

In the previous lecture, we defined serial cost-sharing to be the following: Given shares $q_1 \leq q_2 \leq \ldots \leq q_k$, let player i pay cost $\varsigma_i = \frac{1}{k-i+1}[C(q_1 + \ldots + q_{i-1} + (k-i+1)q_i) - \sum_{j=1}^{i-1} \varsigma_j]$.

With the above assumptions, we showed last lecture that the serial cost-sharing mechanism yields a deterministic Nash equilibrium that is unique, and easy to derive.

2 Quality of cost sharing & Pareto Optimality

As in previous lectures, once we have a Nash solution, it is reasonable to consider the quality of the Nash. It turns out that even under a very loose definition of optimality, the serial-cost sharing solution fails to be optimal. However, this isn't a particular property of serial cost-sharing, as it turns out that no matter the cost sharing mechanism one uses, it will have a Nash that is not optimal. The goal of the remaining part of the lecture is to define a notion of quality, Pareto optimality, and to prove the following theorem, by Moulin & Shenker:

Theorem 1 *(Moulin,Shenker) (Sketch) All cost sharing mechanisms result in utilities that are "not so good".*

To make the notion of goodness more rigorous, we have the definition:

Definition 1 *A solution, $(q_1, \ldots, q_k), (\varsigma_1, \ldots, \varsigma_k) \geq 0$ such that $\sum_{i=1}^{k} \varsigma_i = C(q)$, $q = \sum_{i=1}^{k} q_i$ is* Pareto-optimal *if there is no other solution $(q'_1, \ldots, q'_k), (\varsigma'_1, \ldots, \varsigma'_k)$ such that $u_i(q_i, \varsigma_i) \leq u_i(q'_i, \varsigma'_i)$ for all users i and there is some user j such that $u_j(q_j, \varsigma_j) < u_j(q'_j, \varsigma'_j)$.*

Note that this definition only considers orders of utilities, where it only matters if one is greater than the other, but doesn't distinguish between an outcome with a slightly higher utility versus an outcome with a much higher utility.

Theorem 2 *(Moulin,Shenker) (2^{nd} version) There is no cost-sharing mechanism that yields a Pareto-optimal outcome.*

To actually prove this theorem, we will apply some simplifying assumptions. First, we will relax the strict concavity condition on the user utilities, and say that the utility of user i is $u_i(q_i) - \varsigma_i$, where u_i is a strictly concave, differentiable, and monotone increasing function on the quantity. Furthermore, we will assume that ς_i as a differentiable function of the quantities, $(q_1, \ldots, q_k)$. We will prove that no differentiable cost-sharing $\varsigma_1, \ldots, \varsigma_k$, yields a Pareto-optimal outcome.

Theorem 3 *(Moulin,Shenker) (Final version) Given $C(q)$ strictly convex, differentiable, and monotone increasing, every differentiable cost sharing $\varsigma_i(q_1, \ldots, q_k)$ (for all $i = 1, \ldots, k$, such that $\sum_{i=1}^{k} \varsigma_i(q_1, \ldots, q_k) = C(q) = C(\sum_{i=1}^{k} q_i)$) has a Nash that is not Pareto-optimal.*

To prove this theorem, we need the following lemma:

Lemma 4 *For any cost shares ς_i, cost $C(q)$ satisfying all of the same assumptions as in Theorem 3, for any quantities q_i, there exist utility functions u_i that make $(q_1, \ldots, q_k)$ Nash.*

Proof of Lemma: By definition, quantities $(q_1, \ldots, q_k)$ is Nash if given for each player i, and fixing all q_j for $j \neq i$, q_i is the quantity that maximizes $u_i(q_i) - \varsigma_i(q_1, \ldots, q_i, \ldots, q_k)$. Therefore, at the Nash equilibrium, $(q_1, \ldots, q_k)$, $u_i'(q_i) - \frac{\partial}{\partial q_i}(\varsigma_i(q_1, \ldots, q_k)) = 0$. Therefore, for $(q_1, \ldots, q_k)$ to be Nash, it suffices to have $u_i'(q_i) = \frac{\partial \varsigma_i(q_1, \ldots, q_k)}{\partial q_i}$. The left hand side of that equation is just some number, since q_i's and ς_i's are known, so we just need to find u_i with fixed slope at a single point. Clearly, such u_i will always exist. □

Proof of Theorem 3: From the proof of the lemma, we know that for any Nash solution $(q_1, \ldots, q_k)$, each q_i has the property that $u_i'(q_i) = \frac{\partial}{\partial q_i}[\varsigma_i(q_1, \ldots, q_k)]$.

Consider the situation where we fix everyone's quantity and cost share except for user i. Make i increase (or decrease) his quantity q_i by some small amount. Have user i's cost share absorb the entire resulting change in the total cost. If the cost share $(\varsigma_1, \ldots, \varsigma_k)$ is Pareto-optimal, then that change cannot lead to improvement for user i. For sufficiently small increase in quantity, user i's change in utility is $u_i'(q_i)$. His new cost share is $\bar{\varsigma}_i = C(q) - \sum_{j \neq i} \varsigma_j$ (since he absorbs all of the change in cost), so his change in payment is $C'(q)$. Pareto-optimality means that $u_i(q_i) - C(q)$ is maximized. Therefore, $(q_1, \ldots, q_k)$ is Pareto-optimal $\Rightarrow u_i'(q_i) = C'(q)$ for all i.

Suppose $(\varsigma_1, \ldots, \varsigma_k)$ yields a Pareto-optimal Nash. Then, we have $u_i'(q_i) = \frac{\partial}{\partial q_i}\varsigma_i(q_1, \ldots, q_k)$ and $u_i'(q_i) = C'(q)$. Therefore,

$$\frac{\partial}{\partial q_i}\varsigma_i(q_1, \ldots, q_k) = C'(q)$$

This is independent of the utilities, so, by the lemma, we can vary user i's quantity over $0 \to q_i$ in the expression, and integrate:

$$\int_0^{q_i} C'(q_1 + \ldots + q_{i-1} + r + q_{i+1} + \ldots + q_k)dr = \int_0^{q_i} \frac{\partial}{\partial r}\varsigma_i(q_1, \ldots, q_{i-1}, r, q_{i+1}, \ldots, q_k)dr$$

By the Fundamental Theorem of Calculus, and the fact that if $q_i = 0$, $\varsigma_i(q_1, \ldots, q_i, \ldots, q_k) = 0$, this gives us

$$C(q) - C(q - q_i) = \varsigma_i(q_1, \ldots, q_i, \ldots, q_k)$$

Since the ς_i's are cost shares, $C(q) = \sum_{i=1}^{k} \varsigma_i(q_1, \ldots, q_k)$. Let $q_i = \frac{q}{k}$. By the lemma, this gives a Nash equilibrium. Then for each i, $C(q) - C(\frac{k-1}{k}q) = \varsigma_i(\frac{q}{k}, \ldots, \frac{q}{k})$. Summing over i, we get:

$$\begin{aligned} kC(q) - kC(\frac{k-1}{k}q) &= C(q) \\ (k-1)C(q) &= kC(\frac{k-1}{k}q) \\ \frac{k-1}{k}C(q) &= C(\frac{k-1}{k}q) \end{aligned}$$

This implies that C is linear, which means it violates our condition of strict concavity $\Rightarrow$ $(\frac{q}{k}, \ldots, \frac{q}{k})$ cannot be Pareto-optimal. $\square$

In particular, since the serial cost-sharing mechanism yields a unique Nash, that Nash must not be Pareto-optimal.

3 Reference

Most of the content of this lecture was drawn from:

Herve Moulin & Scott Shenker, *Serial Cost Sharing*, Econometrica, vol. 60, no. 5, pp. 1009-1037, 1992.

CS 684 Algorithmic Game Theory	April 7, 2004
Lecturer: Eva Tardos	*Scribe: Davina Kunvipusilkul*

For the last two lectures, we consider cost-sharing with *convex* cost function $C(q)$ where $q_1, \ldots, q_k$ are users' desired quantities of goods/services and $q = \sum q_i$. Recall that with convex cost functions, the total cost increases faster than linear and users are worse-off when there are more of them.

Today, we will consider cost-sharing with *concave* function.

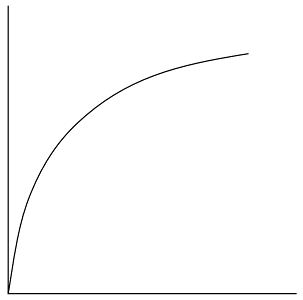


Figure 1: An example of a concave cost function.

In this scenario, users actually like being together because of economy of scale. In addition, users have utility for being included. We will assume that users are inflexible and will only be satisfied if they receive their desired quantities.

Formally, the utility of user $i = \begin{matrix} u_i & \text{if user } i \text{ gets } q_i \\ 0 & \text{otherwise} \end{matrix}$

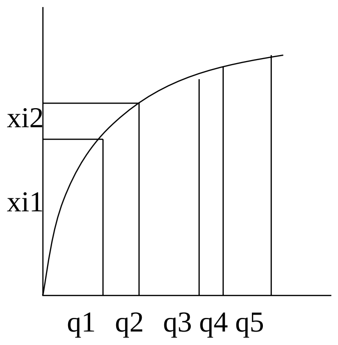
That is, unless the quality is good enough, they don't want to participate.

Recall that we let ξ_i be the cost-share (monotone) for user i. We want to have our cost-sharing scheme to be in the core. That is, we would like to cover the cost $\sum \xi_i = C(N)$ where $C(S)$ is the cost for a set $S \subseteq N$. In addition, users should not be able to do better by splitting into groups, i.e. $\sum_{i \in S} \xi_i \leq C(S)$ all subsets S.

Theorem 1 *Supposer there are k users requesting $q_1, \ldots, q_k$. Let $q = \sum q_i$ and let the cost $C(q)$ be concave. Then cost sharing in the core is possible.*

Proof: Will follow later. □

Consider the following payment scheme where each user pays the additional cost to the system resulting from her or her request.



Making player i pay $\xi_i = C(q_1 + \ldots + q_i) - C(q_1 + \ldots + q_{i-1})$ is in the core. Thus, the Shapley value is in the core.

We will now consider the property of set function $C(S)$ that implies the Shapley value is in the core.

- *Submodularity:* For all sets A and B,

$$C(A) + C(B) \quad \geq \quad C(A \cup B) + C(A \cap B) \tag{1}$$

  (Submodularity is the set system analog of concavity.)

- Equivalently, for all i and sets $A \subseteq B$,

$$C(A+i) - C(A) \quad \geq \quad C(B+i) - C(B) \tag{2}$$

  That is, it is better for user i to arrive later.

Lemma 2 *The two inequalities are equivalent.*

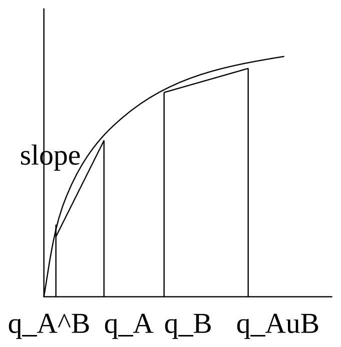
Proof: (1) $\Rightarrow$ (2): Take set $(A+i)$ and B. The only relevent case is when $i \notin B$.
By (1), $C(A+i) + C(B) \geq C(A) + C(B+i)$.

(2) $\Rightarrow$ (1): By induction on $|A \setminus B|$.
(2) is the condition when $|A \setminus B| = 1$.
(The rest is not hard to do.) $\square$

Theorem 3 *If C is submodular, then cost shares $\xi_i = C(1, \ldots, i) - C(1, \ldots, i-1)$ is in the core (and also Shapley value).*

Proof: See later. $\square$

Note: If Cost$(S) = C(\sum_{i \in S} q_i)$ and C is concave, then Cost(S) is submodular.



Consider sets A and B.
$q_A = \sum_{i \in A} q_i$ and $q_B, q_{A\cup B}, q_{A\cap B}$ are similarly defined.
By concavity, the slope $\frac{C(q_A)-C(q_{A\cap B})}{q_A - q_{A\cap B}} \geq \frac{C(q_{A\cup B})-C(q_B)}{q_{A\cup B}-q_B}$.
Note that $q_A - q_{A\cap B} = q_{A\setminus B} = q_{A\cup B} - q_B$.
Rearranging the terms and we get the desired result.

Goal: A game such that

- users tell utility
- design solution share cost
- user may leave if $\xi_i > u_i$
- then new round

By revelation principle, we may assume that users' role is to tell u_i to the algorithm. Note, however, that people may not know their utility.

Question: Are users always truthful? Do they announce true u_i or do they leave when $\xi_i > u_i$?

Well, it depends on the algorithm!

The following theorem gives a property of cost-share and how to deal with leaving players that causes truthful behavior.

Theorem 4 *If the share $\xi_i(S)$ for set S of players is monotone increasing as S decreases, then rational users are truthful. That is, for $i \in A \subset B$, $\xi_i(A) \geq \xi_i(B)$ (**cross-monotone**).*

Theorem 5 *If $C(S)$ is submodular then $\xi_i = C(1,\ldots,i) - C(1,\ldots,i-1)$ and the Shapley value are cross-monotone.*

Proof: We need $\xi_i(A) \geq \xi_i(B)$ for $A \subset B$.
Let $S_i = \{1,\ldots,i\}$.
$\xi_i(A) = C(S_i \cap A) - C(S_{i-1} \cap A)$
$\xi_i(B) = C(S_i \cap B) - C(S_{i-1} \cap B)$
This follows for submodularity with sets $S_i \cap A$ and $S_{i-1} \cap B$. □

Claim 6 *Cross-monotone cost sharing is in the core.*

Proof: $\sum_{i \in S} \xi_i(N) \leq \sum_{i \in S} \xi_i(S) = C(S)$
The inequality follows from cross-monotonicity and the equality from cost sharing. □

CS 684 Algorithmic Game Theory	**Scribe: Prakash Linga**
Instructor: Eva Tardos	**9th April, 2004**

In this lecture we are going to talk about cost sharing in graphs. This is the last lecture on cost sharing. This class is about using primal-dual method for generating cross-monotone shares. Examples of this include:

- Spanning tree (easy version, handled today)
- Submodular functions (not handled in class, refer relevant papers)

Spanning tree as cost sharing

0.1 Problem definition

The problem setting is as follows. Clients want to connect to the server. Here, bandwidth is not an issue and hence clients can connect through each other to finally connect to the server. We are given:

- Server r (root)
- Set of players N (players, clients, users are used synonymously)
- Distances (or costs) on $r + N$ ($r + N \subseteq$ *nodes in G* or $r + N \subseteq$ *metric space*)
- User i has utility U_i

Problem: Select subset $S \subseteq N$ of users, build a min-cost network graph on $S + r$ and share costs between users in S

One interesting special case of a min-cost graph that we consider is the minimum spanning tree (MST). Let $c(S)$ denote the min-cost graph on $S + r$. Also, let $c_{MST}(S)$ be the MST cost on $S + r$.

0.2 Fact

$$c_{MST}(S) \geq c(S) \geq \frac{1}{2} c_{MST}(S)$$

$c_{MST}(S) \geq c(S)$ is obvious. $c(S) \geq \frac{1}{2} c_{MST}(S)$ is a well known result.

0.3 Facts from last class

1. Cost sharing is cross-monotone implies cost shares in core.
2. Cross-monotone cost sharing leads to a strategy proof way to select users and share cost.
3. Cost is submodular.

0.4 Properties of tree cost

We will show that tree cost is sub-additive: If A, B are two disjoint sets then, $c(A)+c(B) \geq c(A+B)$. Note that sub-additive property is necessary for cross-monotone cost sharing.

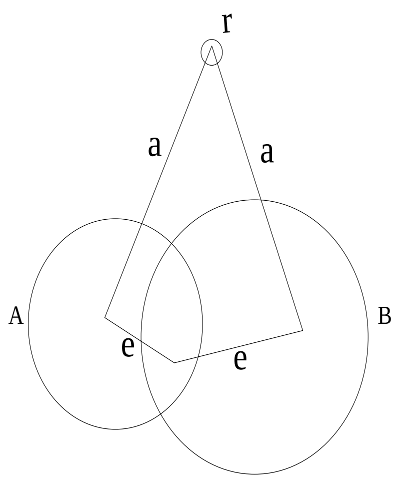


Figure 1: Lemma 2 illustration

Lemma 1

If cost function c admits a cross-monotonic cost-sharing than it must be subadditive. **Proof**. Let $\zeta_i(A)$ be the cost-share of user i in set A. Then we have that $c(A) + c(B) = \Sigma_{i \in A}\zeta_i(A) + \Sigma_{i \in B}\zeta_i(B)$ $\geq \Sigma_{i \in A \cup B}\zeta_i(A \cup B)$ (using cross-monotonicity of ζ_i).

Now, $\Sigma_{i \in A \cup B}\zeta_i(A \cup B) = c(A \cup B)$ (by definition) and hence $c(A) + c(B) \geq c(A + B)$. ■

Lemma 2

Tree cost is sub-additive i.e if A, B are two disjoint sets then, $c(A) + c(B) \geq c(A + B)$

Proof. As the two trees connecting A and B respectively, is a graph connecting $A \cup B$, possibly not the cheapest one. ■

Note

However that $c(S)$ and $c_{MST}(S)$ are not submodular.

Proof.

Considering the graph in Figure 1, $c(A) = a + \epsilon$, $c(B) = a + \epsilon$.

Also, $c(A \cup B) = a + 2\epsilon$. We can choose $c(A \cap B) > a$ so that $c(A \cup B) + c(A \cap B) > c(A) + c(B)$ and hence c is not always submodular. ■

We start by defining a simple cost-sharing. Given node i let $p(i)$ denote the parent of i in the spanning tree. Note that $p(r) = r$. Define cost share $\zeta_i(i)$ as the the cost of edge between i and $p(i)$.

Lemma 3

ζ_i is in core

Proof.

Let's assume ζ_i is not in core. Then let T_s be a cheaper tree on $S + r$.

Now replace edge $(i, p(i))$ for $i \in S$ by T_s. This leads to a cheaper tree. A contradiction.

■

0.4.1 Goal: Want cross-monotone shares for c_{MST}

Refer: Jain and Vazirani; Kent and Skorin-Kapov. The simpler analysis here due to Martin Pal.

0.5 Approximate computation

Note that it is NP-complete to compute $c(S)$ and that $c_{MST}(S)$ is an approximation to $c(S)$ within a factor of 2.

Our goal in this part is to compute cost shares for $c_{MST}(S)$ that are cross-monotone.

Definition ζ_i's are cost shares if $\Sigma_{i \in S} \zeta_i(S) \leq c(S)$

0.5.1 Computation of MST cost shares

We construct an MST using Kruskal's algorithm. The algorithm will maintain sets of nodes and each set will have a "reserve" r. The components will be connected sets. We will think of the reserve r as drawing a ball of radius r around the component. Throughout the algorithm all components will have the same reserve.

- Start with each node in its own component, and $\zeta_i = 0 \ \forall i$, and $r = 0$. Think of r as the reserve "cash" of components, while ζ_i will be the cost-share of node i.
- While not all nodes are connected.
  - Increase component reserve r at the same rate for all components i. When $2r = c(i, j)$ (i.e when the balls of radius r around i and j touch) for two nodes i and j in separate components, build an edge, and merge the components containing i and j to a single component.

    `Charging scheme`: Use the reserve r of one component to "pay" for edge cost. Note that we only pay half the edge cost. The reserve of the other component is kept as reserve for the component. This maintains the invariant that all component reserves are the same. When one of the merged component is the root, than use the non-root component's reserve to pay for the edge.
  - `Cost-shares`: If connected component of node i has n_i nodes, and it does not contain the root, then increase the nodes cost-share at rate $\frac{1}{n_i}$. Do not increase cost-shares in the root component.

0.5.2 Properties

- Tree we build is a MST. This is because we used Kruskal's to obtain the tree.
- In any non-root component, the sum of all cost-shares equals the current reserve r plus the 1/2 of the total cost of all edges built inside the component.
- Components pay exactly for half the edge cost.
- The "reserve" of the root component is never used. (Which is good, since the nodes in the root component do not have cost-shares to pay for this!)

- Shares are cross-monotone. To see this note the following. More users implies rate at which ζ_i increases is slower. Components are connected components when everyone reaches out to some ζ distance. More users therefore implies growth stops earlier and hence the shares are cross-monotone ($\zeta_i(A) \geq \zeta_i(B) \forall A \subset B$).

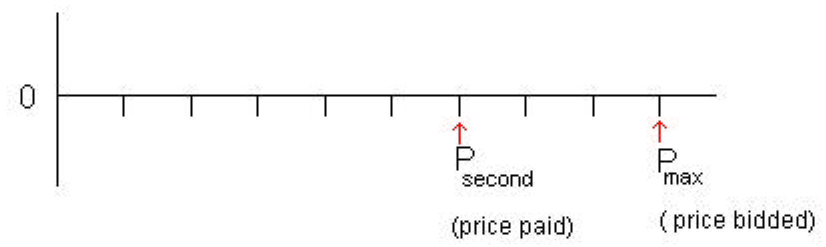
CS 684: Algorithmic Game Theory **Scribe: Jignesh Parmar**
Instructor Eva Tardos **Mon, Apr 12, 2004**

Auctions and Cost Sharing

There are several types of single item auctions:

- Simple auctions
- Increasing/Decreasing price auctions
- First price auctions
- Second price auctions

Today, we consider second price auctions (also called Vickrey auctions named for the Nobel prize economist William Vickrey). In second price auctions, the winner is the person with the highest bid, but the amount that he pays for the item is the second highest price bidded.



Second price auctions are truthful. That is, no matter what others do, one cannot gain profit by announcing incorrect value for the items. In second price auctions, if user **i** has a utility $\mathbf{u_i}$ for an item, and is asked to pay price **p,** the resulting benefit = $\mathbf{u_i - p}$

(N.B. Second price auctions are not group-strategy proof. A group of conspirators can fool an auctioneer)

Cost sharing in VCG auctions is represented as follows:

N represents the users in the second price auction.

For each $S \subseteq N$, $C(S) \geq 0$ (cost is monotone increasing)

Each user $i \in N$ has benefit u_i for item being bidded on.

In the previous two lectures, c(s) was submodular and we discussed the minimum cost spanning tree. The goals were:

Select $S \subseteq N$ and give $x_i(s)$ cost share

1) $\sum_{i \in s} x_i - c(s)$

2) Cross monotone $x_i(A) \leq x_i(B)$, for $A \geq B$

In these cases, there was a group-strategy proof way to select users s and cost shares $x_i(s) \leq u_i$.

Another Goal is to maximize social welfare. Social welfare is defined as

$$\max_{s} \sum_{i \in s} u_i - c(s)$$

Social welfare is not maximized using the previous Shapley value or the Primal-Dual mechanisms. To maximize social welfare, we need to use an algorithm proposed by Vickrey, Clark and Groves (VCG) to select a subset S and to assign cost shares. (N.B. VCG is not budget-balanced)

In VCG, the overall goal is to maximize

$$\max_{s} \sum_{i \in s} u_i - c(s)$$

but the individual goals can be characterized as follows:

$$\text{Max } u_i - x_i(u_1, u_2, \ldots, u_k)$$

$$\underset{\hat{u}_i}{\text{Max}} \; u_i d_{s,i} - x_i(u_1, u_2, \ldots \hat{u}_i \ldots, u_k)$$

$$d_{s,i} = \begin{cases} 1 & \text{if } i \in s \\ 0 & \text{otherwise} \end{cases}$$

In other words, for each item i that is included in the set s, the individual wants to maximize his utility minus his cost share.

VCG Prices for the items can be set as follows

$$i \in s,\ x_i = \quad c(s) - \sum_{\substack{j \in s \\ j \neq i}} u_j$$

$$i \notin s,\ x_i = -\left[\sum_{j \in s} u_j - c(s) \right]$$

Theorem: We state that the VCG mechanism is truthful, and that no individual can gain an advantage by claiming an incorrect utility.

(*N.B.* remember, when truthful, all players want to maximize: $\underset{s}{\text{Max}} \sum_{i \in s} u_i - c(s)$)

Proof:

Assume that player i announces utility $\hat{u}_i$ which is not the same as u_i
He then gets $\sum_{j \in s} u_j - c(s)$ if he is not included in the set S.

If he is included in S , then we select $i \in s$, and he still gets $\sum_{j \in s} u_j - c(s)$

Either way his true benefit does not change, regardless of whether he announces his real utility or not.

CS 684 Algorithmic Game Theory **Scribe: Ashwin Machanavajjhala**
Instructor: Eva Tardos **April 14, 2004**

1 VCG : Vickrey Clarke Groves Mechanism

In today's lecture, we are going to look at the shortest path problem. Consider a graph $G = (V, E)$ and a cost function $c : E \to \Re$. We want to find the shortest path between two vertices $s, t \in V$. In this setting, we model each edge e having cost c_e as a selfish agent A_e. The *agent cost* is 0 if the edge is not in the selected path and c_e if it is on the selected path. If e is on the selected path, A_e gets a payment of p_e. Since the agent is selfish, the agent A_e tries to maximize $p_e - c_e$.

The social welfare goal is to minimize the total cost of the path chosen; i.e., $\min_{all\ paths\ p} \Sigma_{e \in p} c_e$. Note that money is a vehicle used by the economy to ensure social good. Hence, we want to minimize the true cost of the path and not the payments made in the process of acquiring a path. We can write the social welfare objective $\min_{all\ paths\ p} \Sigma_{e \in p} c_e$ as $c(p)$ where p is the shortest path.

Consider an agent A_e. If the agent gets a payment p_e, the agents goal is to

$$\begin{array}{lll} Maximize & p_e - c_e & \text{if } e \text{ is on the selected path} \\ & p_e & \text{if } e \text{ is not on the selected path} \end{array}$$

Note that if the payment is such that it is independent of the agent cost (which is c_e when e is on the path and 0 when e is not on the path), then the agent will truthfully reveal the real cost of the edge. As a first attempt, if $c(p)$ denotes the true cost of a path, we can have payments:

$$\begin{array}{rll} p_e & = c_e - c(p) & \text{if } e \text{ is on the selected path } p \\ & = -c(p) & \text{if } e \text{ is not on the selected path } p \end{array}$$

In this case, the agent gets a profit of $p_e - c_e = -c(p)$, when e is in the selected path and a profit of $p_e = -c(p)$ when e is not in the selected path p. Though truthful, the above payment scheme does not encourage agents to participate in the path selection. To maintain the truthfulness property and elicit voluntary participation, we could add to a positive bonus b_e to every payment such that b_e is independent of the agent cost. So the payments now become:

$$\begin{array}{rll} p_e & = c_e - c(p) + b_e & \text{if } e \text{ is on the selected path } p \\ & = -c(p) + b_e & \text{if } e \text{ is not on the selected path } p \end{array}$$

We want this b_e to be a value independent of the agent cost for each edge. One such b_e is the true cost of the shortest path p^e avoiding e. Since, p^e is the same as p when e is not on the selected path, we have

$$\begin{array}{rll} p_e & = c_e - c(p) + c(p^e) & \text{if } e \text{ is on the selected path } p \\ & = -c(p) + c(p) & \text{if } e \text{ is not on the selected path } p \end{array}$$

Let us look at this payment scheme. Consider an edge e on the selected path p. By how much can the agent A_e lie about the cost of the edge c_e ? Note that if A_e reports a cost $x_e > c_e + c(p^e) - c(p)$, then e is no longer on the shortest path and will not be chosen. Hence, A_e would report a cost of at most $c_e + c(p^e) - c(p)$. This is exactly the payment made to A_e.

Theorem 1 (Truthfulness) *The shortest path with this payment is truthful.*

Proof. Suppose $e \notin p$. A_e has to report a smaller value for c_e so that a path through c_e becomes the shortest path. Hence, A_e would have a negative benefit.

If $e \in p$, A_e would get all the benefit he can get by being untruthful about c_e even by being truthful about the cost c_e. ∎

There are, however, few issues with this solution which have to be addressed to. The first issue is the computation difficulty of the payments. The second issue is that social welfare ignores payments – the sum total of the payments may be much higher than the cost of the shortest path.

Today, we will address the first issue about the computational difficulty of the payments. Note that if the shortest path has k edges, the payments can be computed using k shortest path computations. Suri and Hershberger [1] show that the payments can be computed using only 2 shortest path computations. We outline the ideas in this result next.

First let us consider the special case where the shortest path between s and t passes through all the vertices in the graph. Consider the shortest path between s and t not containing $e = (u, v)$, i.e., $p^{(u,v)}$. For any x which comes before u on the path, $d(s, x)$, which is the shortest distance from s to x is the same whether e is in the graph or not. Similarly, for any y which comes after v in the path, $d(y, t)$ is the same whether G has e or not. p^e should contain an edge in the cut $\{U, V\}$, where $U = \{x|x \text{ lies on } s\text{-}u \text{ path }\}$ and $V = \{y|y \text{ lies on } v\text{-}t \text{ path }\}$. Hence, finding the shortest path in $G \setminus e$ reduces to

$$d(s,t;\ G \setminus (u,v) = min_{(x,y)\in cut(U,V)\setminus(u,v)} d(s,x) + c(x,y) + d(y,t)$$

To compute the shortest path between s, t omitting each edge on p, compute the above expression for every edge (u, v) on the path. This can be done efficiently using a Fibonacci heap. At each phase, keep only those edges which are in the current cut in the heap. The minimum extracted from the heap would give that (x, y) which minimized the above equation and hence the new path. This gives a running time of $O(nlogn + m)$ (for analysis see [1]).

In a general undirected graph, not all vertices lie on p. Consider the shortest path tree rooted at s. To find $p^{(u,v)}$, consider the two components V_s and V_t formed by the removing edge (u, v) from the shortest path tree, V_s contains s and V_t contains t. For any $x \in V_s$, $d(s, x)$ is the same even in $G \setminus (u, v)$. However, it is not as obvious that similarly for any $y \in V_t$, $d(y, t)$ is the same even in $G \setminus (u, v)$, since removing the edge (u, v) in the shortest path tree rooted at t might not yield the same partition.

Lemma 2 *Let y be a vertex in component V_t in $G \setminus (u, v)$. Then $d(y, t)$ is the same in G and in $G \setminus (u, v)$.*

Proof. The proof is by contradiction. Suppose the shortest path from y to t contains (u, v), then the path should traverse (u, v) in the direction from u to v. (since the shortest path from v to y is fully contained in V_y). Then this shortest path can be considered as a concatenation of the shortest paths between y, v and between v, t; with the first sub-path containing u. But since $y \in V_y$, the shortest path shows tree rooted at s shows that the shortest path between v, y is completely contained in V_y. Since G is undirected, the shortest path between y, v should just be a reversal of the above path. But one contains u and the other does not contain u, which leads to the required contradiction. ∎

We have reduced the problem in the undirected case to the special case where all the vertices are on the shortest path between s and t. So one can use a similar algorithm to solve the problem in undirected graph. When trying to compute $p^{(u,v)}$, consider all the edges between vertices in x, y where $x \in V_s$ and $y \in V_t$ and minimize the cost $d(s,x) + d(x,y) + d(y,t)$.

However, the same kind of reasoning does not seem to hold for directed graphs and it was proved that finding the shortest paths for the vickrey payments requires $\Omega(n(nlogn + m))$ time [2, 3].

References

[1] John Hershberger and Subhash Suri. Vickrey Prices and Shortest Paths: What is an Edge worth? In *Proceedings of the 42nd Symposium on the Foundations of Computer Science, (FOCS 2001).*

[2] John Hershberger and Subhash Suri. Erratum to Vickrey Prices and Shortest Paths: What is an Edge worth? In *Proceedings of the 43nd Symposium on the Foundations of Computer Science, (FOCS 2002).*

[3] John Hershberger and Subhash Suri and Amit Bhosle. On the Difficulty if Some Shortest Path Problems. In *STACS 2003*, pp 343-354.

CS 684: Algorithmic Game Theory **Scribe: Mikolaj Franaszczuk**
Instructor Eva Tardos **Friday, April 16, 2004**

Combinatorial Auctions

There is a set N of items, and players who want to bid on subsets of the items. It's combinatorial because a subset may contain more than one item, and the utility of two or more items together may be more then the sum of the utilities of the individual items.

In the general instance, the input is exponential in size, as each user would have to supply a bid for every possible subset. So instead, each user's input specifies both the subset to bid on, and the price they are willing to pay for it.

Formally, the bidders are "single minded bidders":
N is s a set of **k** items.
There are **n** users. Each user **I** has subset $\mathbf{S_i}$ in **N** and value $\mathbf{U_i}$ for receiving $\mathbf{S_i}$

Note that lying by asking for a superset may help a user, as if he wins he would get the set he wants anyway, as it's included in the superset.

Social Welfare

VCG for this problem to decide who gets what.

Define social welfare: a set of disjoint subsets that maximize total value.
Formally:

$$\max \left(\sum_{i \in I} Ui : \text{set } S_i \text{ for } i \in I \text{ disjoint} \right) = OPT$$

Payment bonus:
P1 = OPT if i is not in I

$P2 = OPT - U_i$ if $i \in I$

This is not an easy optimization problem as it is just like set-packing, which is NP-complete. It can’t even be approximated well, as will be shown later.

Heuristic Algorithm

Suppose you have a heuristic algorithm to optimize this (such as integer programming that is stopped after a fixed amount of computational time). Is it truthful?

Assume that users know what algorithm you are using to optimize. Then they will actually lie in order to help you, by giving you extra input! Their welfare is aligned with your optimization accuracy, so it benefits the users to help you optimize. So a heuristic approximation is not truthful.

A paper by Nison and Ronen describes a 2 phase heuristic algorithm:

1. Users announce **Si** and **Ui** to designer
2. Mechanism announces all requests, as well as the heuristics it is using
3. Users are allowed to offer alternates. **($\underline{S}$, $\underline{U}$)**. Each user proposes alternate sets and utilities for all other users.
4. Run the heuristic algorithm on the original set **(S, U)**, and on all n alternates **$(\underline{S}, \underline{U})_i$** for all i. Return the best result from all of those runs.

Approximations

Can we use an approximation algorithm that given an approximation bound? Yes, but they aren’t very good.

Possible algorithms:

(1) Take a single **Si** with the maximum **Ui**. This is a k-approximation and n-approximation algorithm. The worst case is when a set of n individual users wanting individual items is better than some one user taking the entire set.This algorithm is truthful (user will report actual utilities).

(2) Sort by $\frac{Ui}{|Si|}$ and allocate greedily. This is also an n-approximation. A worst case example is when a user I wants some one item for a high price, but another user with slightly lower density wants the entire set. Since the first user will be serviced first, the 2nd user is denied the entire set and the total value is thus comparatively low.

(3) Sort by **Ui** and allocate greedily. This is also an n-approximation. A worst case example is when some one user wants an entire set, but all the other users want all of the items individually, with utilities lower by just epsilon.

(4) Sort by $\frac{Ui}{\sqrt{|Si|}}$ and allocate greedily.

Theorem: (4) is a $\sqrt{n}$ approximation.

Note: unless P=NP, no $n^{\frac{1}{2}-e}$ approximation is possible [due to Hastad]

Theorem: Any one of the strategies 2-4 above can be made truthful by proper payments.

***Proof*:**

First, note that it is never tempting to lie about Si. Declaring a larger set will still give you your original set, but will put you later in the sorting order for the algorithms. Declaring a smaller set is never good because even if you get it, you don't get the full set you originally wanted. But it can be tempting to lie about the utility. Increasing the utility will put you higher in the sorting order. Decreasing the utility could potentially lower your payment. So how should payments be set to avoid such lying?

Set **Pi = 0** if I is not selected
Set **Pi = min value Ui** such that algorithm includes i with this value.

***Claim*: Resulting procedure is truthful:**

If **i** $\in$ **I**, then user i already gets the lowest possible payment, so there is no incentive to decrease utility. There is no incentive to increase it either, as the user is already guaranteed to be in the set

If **i** is not in **I**, then raising **Ui** high enough to get the user in the set will exceed payment, so it won't be worth it to be in the set, so the user won't make this lie.

CS 684 Algorithmic Game Theory	**Scribe: Chandrashekhar Nagarajan**
Instructor: Eva Tardos	**19th April, 2004**

1 Randomized rounding and truthful auction

We consider the same combinatorial auction setup as in last class. There are k "single minded bidders" $1, 2, 3 \ldots k$ and each bidder i wants a subset S_i of items from a collection of items N: $S_i \subseteq N$. Each bidder i has an utility u_i associated with his set S_i. The bidder pays p_i toward getting the set S_i. His happiness is $u_i - p_i$ if he gets the set or $-p_i$ if he does not get the set.

Note that bidders need not reveal their exact utility. So we saw that the VCG mechanism of pricing gives a social optimal solution. But given the true S_i and u_i values

$$max \sum_{i \in I} u_i \ s.t\ sets\ corresponding\ to\ players\ in\ I\ are\ disjoint$$

is NP-complete and also no good approximation algorithm exists.

But this problem is easy to approximate if we have the sets of high multiplicity. Let the multiplicity of an item j be m_j and there are are totally n different items. Let us assume the following :

- $m_j \geq c \cdot log(n)$ and all the copies are indistinguishable.
- Each user wants atmost one copy of an item.

This problem is also NP-Hard but it is easy to approximate this problem and good approximations exists. One of the algorithm for this is *randomized rounding.*

Let us have $\{0, 1\}$ variables x_i for a bidder i. $x_i = 1$ if i gets the set S_i and $x_i = 0$ otherwise. Consider the LP relaxation of the optimization problem.

$$Max \sum_i u_i \cdot x_i$$

$$\sum_{i:j \in S_i} x_i \leq m_j \ \ \forall\ distinct\ items\ j$$

$$0 \leq x_i \leq 1 \ \ \forall\ bidders\ i$$

The last constraint is the LP relaxation of the integer programming problem as we are allowing the integer variable x_i to take fractional values.

Now the main idea of randomized rounding is as follows: Give the set S_i to the bidder i independently with probability x_i. This makes sense as $0 \leq x_i \leq 1\ \forall i$.

Note that the expected sum of utilities $= E(\sum_{i \in I} u_i) = \sum_i u_i \cdot x_i$ since expectation is linear. Note that $\sum_i u_i \cdot x_i \geq$ Social Optimum as the integer optimum is also a solution to the LP and we are maximizing over fractional solutions also. Also note that

$$E(\#\ copies\ of\ an\ item\ j\ sold) = \sum_{i:j \in S_i} x_i \leq m_j$$

To keep the # of each item sold well under control we decrease the probability of giving the set S_i to the bidder i from x_i to $(x_i/1+\delta)$.

So now

$$E(\sum_{i \in I} u_i) = \sum_i \frac{u_i \cdot x_i}{1+\delta}$$

$$E(\#\ of\ copies\ of\ item\ j\ sold) = \sum_{i:j \in S_i} \frac{x_i}{1+\delta} \leq \frac{m_j}{1+\delta}$$

If $Y_1, Y_2, \ldots Y_n$ are independent $\{0,1\}$ variables with $E(\sum Y_i) \leq m$ and $\delta > 0$ then by Chernoff Bounds

$$Prob(\sum Y_i > (1+\delta)m) \leq \left(\frac{e}{(1+\delta)^{(1+\delta)}}\right)^m$$

Applying this result here,

$$Prob(\sum_{i:j \in S_i} x_i > m_j) \leq \left(\frac{e}{(1+\delta)^{(1+\delta)}}\right)^{\frac{m_j}{1+\delta}}$$

We can select c and δ such that this probability is less than ϵ/n.

Corollary 1 *Probability that there is an over sold item is less than or equal to ϵ.*

This is because there are atmost n items and probability that each item is oversold is $\leq \epsilon/n$. ■

So if there is an oversold item during the auction, cancel the auction and rerun it once again. So with probability ϵ the auction gets canceled. So we can use this procedure to implement truthful auction.

Now we shift our attention to bidders/users. Users are risk neutral and they want to maximize their expected happiness. Let user i gets his set S_i with probability y_i. The user pays the price p_i and his expected profit $= u_i y_i - p_i$. So every user tend to maximize $u_i y_i - p_i$.

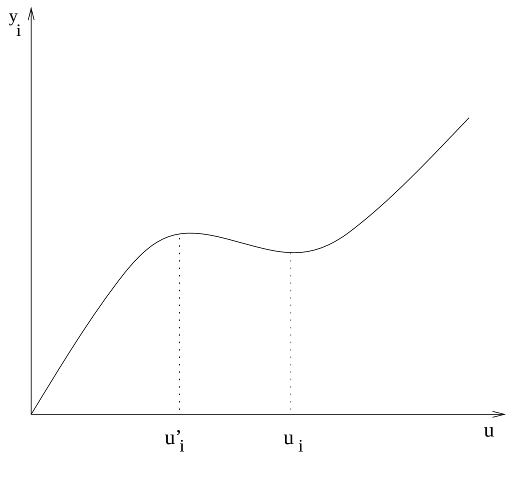


Figure 1: Non-monotone probability function

Let us see what happens if the probability function is not monotone increasing on the utility stated u_i. Consider the case in Figure 1.

If a player i's original utility is u_i and if the probability of getting his set increases when he lies about his utility to $u'_i < u_i$ then the player can increase his chance of getting the set by lying about his utility. We claim that this makes the auction untruthful. To see this consider the expected price p_i and p'_i that the user has to pay in the two cases above. If $p'_i \le p_i$, then the user would increase his change of getting the item and decrease his cost by claiming utility u'_i when the true utility is u_i, so the auction is not truthful. So we must have $p'_i > p_i$. Now assume the user's true utility is u'_i, then the expected benefit of user i claiming utilities u_i and u'_i respectively is $u'_i y_i - p_i$, and $u'_i y'_i - p'_i$. For the auction being truthful, we must have that $u'_i y_i - p_i \le u'_i y'_i - p'_i$, and hence $p_i - p'_i \ge u'_i(y'_i - y_i) > 0$, a contradiction.

Theorem 2 *If y_i is a monotone increasing probability function of utility of player i which denotes the probability of the player i getting his set and the price he pays is $p_i = u_i y_i(u_i) - \int_0^{u_i} y_i \, du_i \ \forall i$ then the players will behave truthfully.*

Proof:

A player i has expected profit $= \int_0^{u_i} y_i \, du_i$ if he says his utility truthfully.

Case 1: Let a player i lie about his utility as $u'_i > u_i$.

So the probability of i getting the set S_i is $y_i(u'_i)$.

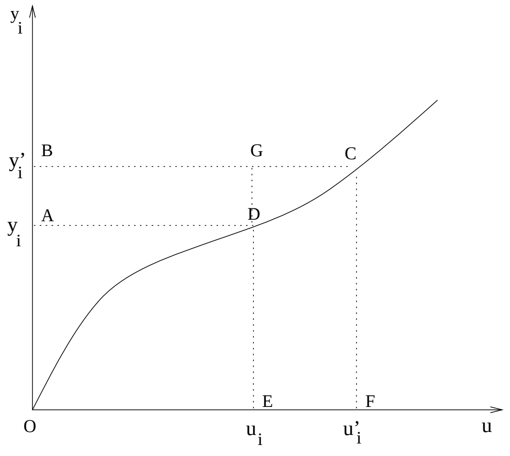


Figure 2: Case 1: $u'_i > u_i$

$$\begin{aligned}
Expected\ profit &= u_i y_i(u'_i) - (u'_i y_i(u'_i) - \int_0^{u'_i} y_i \, du_i) \\
&= \int_0^{u'_i} y_i \, du_i - (u'_i - u_i) y_i(u'_i) \\
&= \int_0^{u_i} y_i \, du_i - Area(GDC) \quad (from\ Figure\ 2) \\
&\le \int_0^{u_i} y_i \, du_i \quad (As\ y_i\ is\ monotone\ increasing)
\end{aligned}$$

So the bidder has no incentive to lie by increasing his utility.

Case 2: If the player i lies about his utility as $u'_i < u_i$.

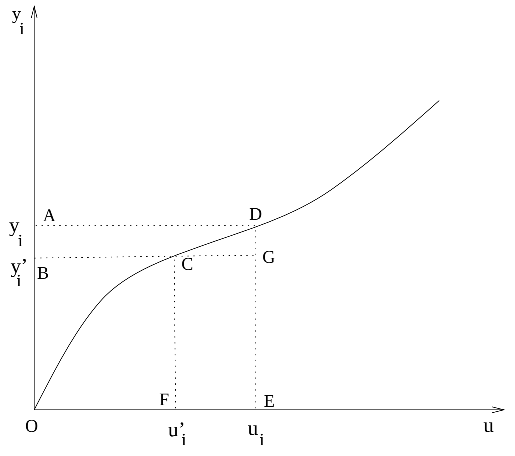


Figure 3: Case 2: $u'_i < u_i$

$$\begin{aligned}
Expected\ profit &= u_i y_i(u'_i) - (u'_i y_i(u'_i) - \int_0^{u'_i} y_i\ du_i) \\
&= \int_0^{u'_i} y_i\ du_i + (u_i - u'_i) y_i(u'_i) \\
&= \int_0^{u_i} y_i\ du_i - Area(CGD) \quad (from\ Figure\ 3) \\
&\leq \int_0^{u_i} y_i\ du_i \quad (As\ y_i\ is\ monotone\ increasing)
\end{aligned}$$

So the player i has no incentive to lie as his expected profit only goes down if he lies. So this setting is truthful if y_i is monotone increasing with u_i. ■

Note that x_i which is a result of solving the LP relaxation is monotone increasing with the utility stated u_i, and so $(x_i/1 + \delta)$ is also monotone increasing. But actually there is a slight problem as by decreasing u_i a player can affect the probability of over sold item and so can affect the probability of cancelling the auction which may be an incentive to lie. To get around this difficulty the auctioneer can cancel the auction with a fixed probability ϵ. This makes the auction truthful.

CS 684 Algorithmic Game Theory	Scribe: Prakash Linga
Instructor: Eva Tardos	23rd April, 2004

In today's class we will start to talk about the limits of what can be done truthfully in auctions. We will see that VCG is the only possible algorithm in many cases.

1 Road map

- `Arrow's impossibility theorem on voting` : Today's class
- `Gibbard-Satlerthwaite` : Impossibility of truthful mechanisms without money transfer.
- `Robert's` : In a certain general model, only truthful mechanism is weighted VCG.

2 Arrow's Impossibility Theorem

2.1 Problem

There are N users (players or voters). Each player has an ordered list of preferences ($x > y > z > \cdots$) on ≥ 3 candidates (or resources). Can we select a winner such that

1. `Unanimity preserving`: If all players have $x > y$ then y does not win.
2. `Independence of irrelevant alternatives or Pair-wise preference relationship`: Given some preferences, let's say the result is x and not y. A change in preference of z will not cause y to win.
3. `Non-dictatorial`: The solution will not be just copying a certain player's list.

2.2 A simple algorithm proposal

Define $x > y$ if majority prefers x over y.

This is not good enough because of `Condescet paradox`. Consider three users with the following preference lists:

- User 1: $a < b < c$
- User 2: $b < c < a$
- User 3: $c < a < b$

From the above preference lists, $a < b < c < a$ and hence it is not possible to select a winner.
`Corollary`: With ≥ 3 candidates there is no truthful election scheme.

2.3 Theorem

There is no selection rule satisfying the above three properties

We will show that any selection rule satisfying rules 1 and 2 is dictatorial.

Now, any selection rule satisfying rules 1 and 2 can be extended to generate a total order $>$ of candidates such that

- All players prefer x to y implies $x > y$.
- Outcome of $x > y$ depends only on users preference between x and y.

How to generate a total order on candidates

Define overall $x > y$ as follows: Given user i preference $a > \cdots > x > \cdots > y > \cdots$, change this to put x and y at the begining. The modified preference list would be $x > y > a > \cdots$.

- **Claim**: x or y wins.

  Proof. Follows from axiom 1. ■

  We will define that $x > y$ overall if x wins.

- **Claim**: Above definition of $>$ is an order i.e $x > y \wedge y > z \Rightarrow x > z$

  Proof. Move x, y to the top two places. Now x wins by definition. Now slide y to its place among the the top 3.

  We claim that the winner is now x or z and not y. This follows from axion 2.

  If x wins then $x > z$ (i.e decreasing y is irrelevant) and this is what we want to prove.

  If z wins then $z > y$ (i.e decreasing x is irrelevant). A contradition. ■

Definition: Given that $x > y$, a subset S is *decisive* if all $i \in S$ prefer $x > y$ and all $i \notin S$ prefer $y > x$ implies outcome is $x > y$.

Note that outcome only depends on x over y choices, based on axiom 2.

Lemma: S decisive for $x > y$ then S is also decisive for all $a > b$.

Proof. We will show that: S decisive for $x > y$ implies also decisive for $x > b$.

x, b preference does not depend on y. Therefore, choose $y > b$. Consider the following setup:

- $S : x > y > b$
- $\bar{S} : y > b > x$

$x > y$ in S and $y > x$ outside S, so we know that $x > y$ overall (as S is decisive). Also we chose $y > b$ everywhere, so $y > b$ by axiom 1. Therefore $x > b$ by transitivity. Now $x > b$ in S and $b > x$ outside of S, so S is decisive for $x > b$.

Choose $x > a$ and use a similar argument as above argument with $x > b$ to get S is decisive on $a > b$ ■

Observation: S or $\bar{S}$ is decisive for any S.

Lemma: S and S' decisive implies $S \cap \bar{S}$ is decisive.

Proof. To prove this consider 3 elements x, y, and z. Assume that we have the following:

- $S \setminus S'$: $x > z > y$ y
- $S \cap S'$: $x > y > z$
- $S' \setminus S$: $y > x > z$
- $S \bar{\cup} S'$: $z > y > x$

Now S is decisive, so we get $x > y$. Also S' is decisive, so we get $y > z$. so by transitivity, we get that $x > z$. But now notice that $x > z$ is only true on $S \cap S'$ so we conclude that $S \cap S'$ is decisive.

■

Finally, proof of Arrow's theorem

Proof. (**Arrow's theorem**)

Let $v \in N$ be some voter. Then either $\{v\}$ or $N - v$ is decisive. If none of $\{v\}$ for some $v \in N$ is decisive then $\cap_v N - v$ is decisive. But $\cap_v N - v = \phi$, the empty set. This contradicts axiom 1 (unanimity preserving).

If for two different nodes v and w the sets $\{v\}$ and $\{w\}$ are decisive, than their intercestion is decisive, which is empty. We get the same contradiction.

So it must be the case that exactly one set $\{v\}$ is decisive, and for all other $w \neq v$ the set $N - w$ is decisive. But then all sets containing v are decisive (as they are intersections of sets of the form $N - w$). This means that v is the dictator.

■

Note that Arrow's impossibility theorem is a weak result because there are no weights and no bribes.

CS 684 Algorithmic Game Theory	**Scribe: Christina Tavoularis**
Instructor: Eva Tardos	**Monday April 26th, 2004**

1 Gibbard-Satterthwaite Theorem

We will begin by recalling an important result that was introduced in the last lecture. For an election with at least 3 candidates and where each voter has an ordered preference for the candidates, consider two claims:

- *Claim 1: Unanimity Preserving* If $x > y$ and candidate x is preferred by all voters then y is not selected.
- *Claim 2: Independent Irrelevant Alternatives* If x is selected on some preferences, and the preference of alternative z is changed, then that cannot cause $y \neq x, z$ to be selected, although z may be selected as a result.

Arrow's Theorem states that when players are truthful[1], claims 1 and 2 are only satisfied in a dictatorial system. Recall that in a dictatorial system, the outcome is based on one players highest preference. Now we can present a stronger result, courtesy of Gibbard and Satterthwaite. Assume we have a set X of candidates and the goal is to select exactly 1. Voters have utilities $u_i(x) \geq 0$ for each candidate $x \in X$ such that the utility function encodes users' preferences: $u_i(x) > u_i(y) \sim x > y$. Furthermore, we will assume that side payments (bribes) are not allowed.

When there are only 2 candidates, it is clear that there is no harm to admitting one's preferences. In this case, the truthful election is simply majority voting. A voter against the majority can only change her strategy by voting with the majority which has no benefit. The following theorem tells us that we cannot make the same inference for more than 3 candidates.

Theorem 1 (Gibbard-Satterthwaite) *No truthful election system can be achieved when there are* 3 *or more candidates and where candidates can be elected by user preferences.*

Example Assume we have three candidates x, y, z. A truthful system can be achieved if it is known in advance that candidate z cannot get elected. In this system, those whose first preference is z can use their second preference.

Before formally proving the theorem, we will comment on the monotonicity of any individual utility function. Assume candidate x is selected. Monotonicity implies that for some user i if $u_i(x)$ increases, or $u_i(y)$ decreases for $y \neq x$, then the outcome does not change.

Claim Truthful $\Rightarrow$ monotone.

Proof. Suppose $u_i(x)$ increasing to $u'_i(x)$ causes the outcome to change from x to y. If $u_i(y) > u_i(x)$ then player i is happy and should be claiming $u'_i(x)$ even when $u_i(x)$ is the truth. If $u_i(y) < u_i(x)$ then he should stay with his original claim $u_i(x)$ even when $u'_i(x)$ is true. Thus, being truthful implies monotonicity. The same argument can be used for $u_i(y)$ decreasing. $\blacksquare$

[1]Truthfulness requires that every voter (i.e. player) cannot express untruthful preferences.

Proof. (of Gibbard-Satterthwaite) Assume candidate x has been selected. First we wil prove the independence of irrelevant alternatives. For some i, assume i decreases its utility for candidate z. By monotonicity, x will remain the outcome. But what happens if i increases its preference for z from $u(z)$ to $u'(z)$? Suppose that the winning candidate is z when player i uses preference u'. Let $u''_i(z) = min(u_i(z), u'_i(z))$. Under the u preference scheme x is elected, and by monotonicity $\Rightarrow$ x is still elected in u''. However, under the u' system y is elected, and by monotonicity y is elected in u'', contradiction. Thus, both system must elect the same candidate.

Now we will show that this scheme is unanimity preserving. Suppose not. Then, $u_i(x) > u_i(y)$, $\forall i \Rightarrow$ y is elected. For simplicity, let $u_i(z) = 0$ $\forall i$ and $\forall$ $z \neq x, y$ (by monotonicity we can show that we are solving the same problem). This simplifies the game to a two person system. There exists a preference $\hat{u}_i$ that elects x (again we will assume that $\hat{u}_i(z) = 0$ $\forall i$ and $\forall$ $z \neq x, y$). Consider switching players one by one from using u to $\hat{u}$, forming an ordered list. After some finite number of switches, the outcome changes. In the first case, it changes to $z \neq x, y$. This outcome is undesirable for user i because $\hat{u}_i(z) = 0$ and thus user i would never admit such a change. Otherwise, the new outcome is x, in which case the subset of players are better off claiming $\hat{u}$ because x is in fact preferred by all players. ■

2 Roberts Theorem and Related Results

Arrow's and the Gibbard-Satterthwaite theorem are election results. We can think of systems such as combinatorial auctions and network routing as elections. For any algorithm, the candidates are the possible outcomes.[2] The Gibbard-Satterthwaite theorem holds for no side payments and arbitrary valuations of outcomes. We would also like to consider algorithms with no externalities such that players only care about their personal gain and there is no correlation between players preferences. A more general result presented by Roberts allows for arbitrary preferences, and side payments where preferences are linear in the payment amount. As usual, we will think of the problem as an election. Assume that x is elected and user i has preference $u_i(x) + p_i$ where p_i is a payment.

Recall the VCG auction which maximizes the social happiness $\max_x \sum_i u_i(x)$ according to a payment system where player i pays $p_i = \sum_{j \neq i} u_j(x)$. The weighted VCG auction selects x according to $\max_x \sum_i w_i u_i(x) + h(x)$ where $h(x)$ is a penalty. The weighted VCG auction can be implemented truthfully with side payments. User i should receive payment $p_i = 1/w_i[\sum_{j \neq i} w_j u_j(x) + h(x)] - h_i(\text{other users})$. We require however that $h_i(\cdot)$ is such that $p_i \geq 0$.

Theorem 2 (Roberts) *The only truthful outcome with payments is a weighted VCG.*

Proof. The proof of Roberts theorem requires both Arrow's and Gibbard-Satterthwaite theorems. The proof is omitted.

Lavi, Mu'alem and Nisan present an analog of Roberts theorem for combinatorial auction with no externalities in preferences. They provide three axioms:

1. Truthful auction $\Rightarrow$ monotone

[2]Note that in this case there could be exponentially many candidates, however this theorem is not concerned with the complexity of the problem.

2. monotone + independence of irrelevant alternatives $\Rightarrow$ weighted VCG.
3. If preference $u \to a$ and $u' \to b$ then there exists a user i such that $u_i(a) - u_i(b) > u'_i(a) - u'_i(b)$.

Axiom 3 is a stronger result that encodes the linearity of preferences, whereas Roberts only codes preferences according to payoff. However, the general implications are much like those of Gibbard-Satterthwaite.

References

[1] K. Roberts: The characterization of implementable choice rules. In Aggregation and Revelation of Preferences, Laffont, J.-J. (Ed.), North-Holland.

[2] Ron Lavi, Ahuva Mu'alem, Noam Nisan: Towards a Characterization of Truthful Combinatorial Auctions. FOCS 2003: 574-583

[3] Ahuva Mu'alem, Noam Nisan: Truthful Approximation Mechanisms for Restricted Combinatorial Auctions. AAAI/IAAI 2002: 379-384

CS 684 Algorithmic Game Theory | **Scribe: Xin Qi**
Instructor: Eva Tardos | **April 30, 2004**

1 Single item auction with k identical Bayesian users

This section is a follow-up of the last lecture. We have k users (bidders), user i gets value v_i if he wins the bidding. Every user has this value independent with each other, and all v_i are drawn from the *same* distribution. We have a first-price auction, i.e. the user with highest price will win, and will pay whatever he/she declares.

Theorem 1 *Let*

$$\beta(v) = Exp(\max_{j \neq i} v_j | v_j \leq v, \forall j \neq i) \tag{1}$$

then every user i bidding $\beta(v_i)$ is a Nash equilibrium.

Proof. W.l.o.g., we consider user 1, and his/her valuation v.
Let

$$X = \max_{j>1} v_j \tag{2}$$

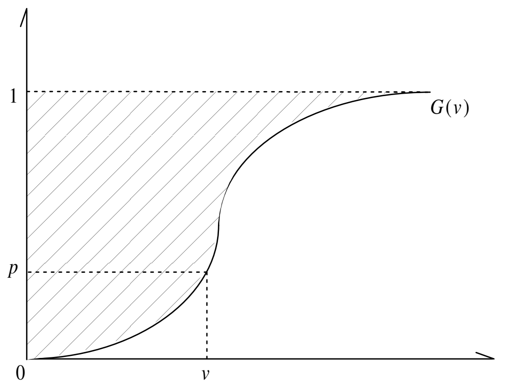
and

$$G(v) = Pr(X \leq v) \tag{3}$$

If we want to calculate the conditional expectation $\beta(v)$ as defined in the theorem, we will need $Exp(X)$ and $G(v)$.

To simplify our proof, let us make the assumption that all the functions (probability, $\beta(\cdot)$, $G(\cdot)$, etc.) have whatever smoothness condition (continuous, differentiable, etc.) we required.

It is easy to see that $G(v)$ is monotone increasing, as shown in the following figure.



From the figure above, we can make the following two claims:

Claim 2

$$\begin{aligned} Exp(X) &= \textit{shaded area} \\ &= \int_0^\infty Pr(X \geq v)dv \\ &= \int_0^1 G^{-1}(p)dp \end{aligned}$$

Claim 3

$$\begin{aligned} & Exp(X|X \leq v) \cdot Pr(X \leq v) \\ = & \int_0^{G(v)} G^{-1}(x)dx \\ = & \; v \cdot G(v) - \int_0^v G(x)dx \end{aligned}$$

Suppose users $j \geq 2$ bid $\beta(v_j)$, then what should user 1 do? Suppose he/she bids b, then his/her happiness is

$$(v_1 - b) \cdot Pr(b \geq \max_{j\geq 2} b_j) = (v_1 - b) \cdot Pr(b \geq \max_{j \geq 2} \beta(v_j)) \tag{4}$$

The goal is to prove that $b = \beta(v_1)$.

First, we need to show that there exists some z s.t. $b = \beta(z)$. The reason for this is that the maximum useful bid for user 1 is $\leq \max_v \beta(v)$, since it is the maximum possible bid of all the other users, and the $\beta(\cdot)$ function is continous.

Second, we need to show that $z = v_1$. Consider the happiness of user 1, and we will obtain the following result by the monotonicity of $\beta(\cdot)$ and the assumption that all the users have the same distribution:

$$\begin{aligned} & (v_1 - \beta(z)) \cdot Pr(\beta(z) \geq \max_{j\geq 2} \beta(v_j)) \\ = & \; (v_1 - \beta(z)) \cdot Pr(z \geq \max_{j \geq 2} v_j) \\ = & \; (v_1 - \beta(z)) \cdot G(z) \\ = & \; v_1 \cdot G(z) - \beta(z) \cdot G(z) \\ = & \; v_1 \cdot G(z) - z \cdot G(z) + \int_0^z G(x)dx \end{aligned}$$

To maximize the happiness of user 1, we take the derivative of the formula above, and let it be zero:

$$\begin{aligned} & v_1 \cdot G'(z) - z \cdot G'(z) - G(z) + G(z) = 0 \\ \Rightarrow & \; v_1 \cdot G'(z) - z \cdot G'(z) = 0 \\ \Rightarrow & \; v_1 = z \end{aligned}$$

■

From above, we know that even if the mechanism is first-price auction, users will pay as if it was a second-price one.

Now we want to generalize this result, i.e. for all the auction mechanisms that satisfy some assumptions, Bayesian users will behave as in a second price auction.

2 Revenue equivalence theorem

First, we need to make several assumptions:

1. In Nash, the person with highest valuation will win. (For identical distributions, this is reasonable).
2. A function $e(p)$, as defined below, is continuous.

   Given all others' strategies, a user will choose a bid b, which affects

   - expected payment, which is not conditioned on winning;
   - probability p of winning.

   Assuming the function from bids b to probabilities p is continuous and invertible, we define $e(p)$ as the expected payment on bid with winning probability p. One of our assumption is that the function $e(p)$ now defined is continuous.
3. The function $e(\cdot)$ satisfies $e(0) = 0$, i.e. if a user does not want to win, he/she will pay nothing.

The user wants to optimize their expected happiness, which is

$$v \cdot p - e(p) \tag{5}$$

where he/she can control probability p. By basic calculus, he/she will choose p s.t. $v = e'(p)$. By assumption 1,

$$p(v) = Pr(v \text{ is the highest}) = G(v) \Rightarrow v = G^{-1}(p) \tag{6}$$

Therefore, we have

$$e'(p) = G^{-1}(p) \tag{7}$$

So

$$e(p) = \int_0^p G^{-1}(x)dx + e(0) = \int_0^p G^{-1}(x)dx \tag{8}$$

So the expected payment of user with value v will be

$$\int_0^{G(v)} G^{-1}(x)dx, \tag{9}$$

which is the same as the bid in the first price auction by Claim 3.

The above result has the hidden restriction that there is always somebody winning the auction, but we can alos allow the auctioneer not to sell when the price is too low.

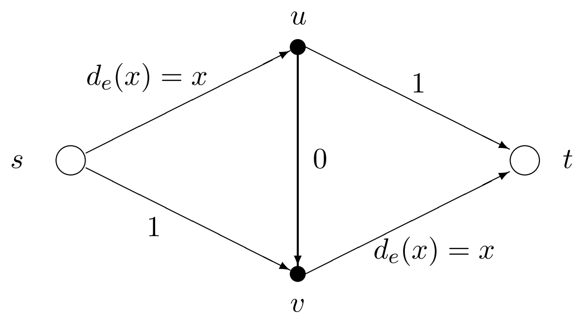
For example, consider the special case that there is only one bidder, then if the auctioneer has to sell, then his/her income will always be zero, no matter what mechanism is in use. If the auctioneer has the option not to sell to anybody, he/she can declare a price r, and sell the item only if the price is higher than r. The income will be $r \cdot Pr(v \geq r)$. Here, r is called "reserve price". This can be viewed as the auctioneer is also a bidder offering a price r.

There is also a stronger version of the revenue equivalence theorem, which basically says that under the assumptions similar to what was used above, the auctioneer's income only depends on the probability he sets for not selling. For example, this means that there is an auction maximizing the auctioneer's income that is a first price (or second price) auction with some reserve price r.

CS 684 Algorithmic Game Theory	**Scribe: Zoya Svitkina**
Instructor: Eva Tardos	**May 3, 2004**

1 How bad is Braess' paradox

Let us review what Braess' paradox is, using the same example as we had before. The setting is a routing game on a graph. The delay $d_e(x)$ on edge e is a function of the amount of traffic x sent on e. We assume that $d_e(x)$ is monotone and continuous. In this example, the traffic rate from s to t is 1.



The cost of a flow in Nash equilibrium is evaluated as the delay on a path from s to t (recall that at Nash equilibrium, all $s - t$ paths have the same delay). In the example, if the edge uv is not present, then the Nash flow is 1/2 on each of the two paths, and the resulting delay is 1.5. However, if the edge uv with zero delay is added to the graph, then the equilibrium flow is 1 on the path $s - u - v - t$, and the delay is 2. Braess' paradox is the fact that the seemingly beneficial action of adding a new edge in fact only increases the delay.

Before, we had a function representing social cost, $\sum_p f_p d_p(f)$, where f_p is flow on a path p, and we studied the "price of anarchy", which is $\frac{\textit{social cost of Nash}}{\textit{opt social cost}}$. Now we will focus on Braess' paradox and study the ratio $\frac{\textit{social cost of Nash}}{\textit{social cost of Nash on best subgraph}}$.

The first observation is that the ratio for Braess' paradox is at most the price of anarchy. This is because routing the flow on a subgraph is one of the things that *opt* can do, so the optimal delay is not more than the best delay on a subgraph. For example, when delays are linear, $d_e(x) = a_e x + b_e$, since the price of anarchy is at most 4/3, so is the Braess' paradox ratio.

Let us try to construct an example with high Braess' paradox ratio. The bad example that we had for the price of anarchy was a two-edge two-node graph with delays x^N and 1 on the two edges, and a traffic rate of 1. In Nash equilibrium, all the flow goes on the first edge, with a delay of 1, whereas the optimum is to send ϵ flow on the edge with delay 1, and the rest on the other edge, for an average delay of $\epsilon \cdot 1 + (1-\epsilon)(1-\epsilon)^N \approx \epsilon$. However, there is no Braess' paradox on this graph (the ratio is 1).

As a second attempt to construct an example with high ratio, let us change the delay function on the variable-delay edges in the picture above from x to x^N. As a result, when the middle edge is absent, the delay at Nash is close to 1, whereas when it is present, the delay is 2.

Here are some facts which are known about Braess' paradox:

1. For a single source-sink pair, the worst Braess' paradox ratio is $n/2$, where $n = |V|$ (Roughgarden 2001).
2. It's NP-hard to find a good subgraph, even if the class of examples is guaranteed to only contain networks with Braess' paradox ratio which is either $n/2$ or 1 (Roughgarden 2001).
3. If the graph contains 2 $s - t$ pairs, then the Braess' paradox can be as bad as $\sim 2^{O(n)}$.

Now we will prove an upper bound of n for the first fact, and will show an example illustrating the third fact. We will consider a single source - single sink network and look at the $s - t$ delay on it.

Theorem 1 *For flow with a single source-sink pair, the Braess' paradox ratio is at most n.*

Let us say that f is a Nash flow that has delay D, and f^* is Nash flow on the best subgraph, with a delay of D^*. For some edge e, if $f^*(e) > 0$ and $f(e) \leq f^*(e)$, then we will call such an edge *good*, which means that it is not deleted and is part of the best subgraph.

Claim 2 *If e is good, then $d_e(f) \leq D^*$.*
Proof.

$$d_e(f) \leq d_e(f^*) \leq D^*$$

where the first inequality follows from monotonicity of d_e, and the second one follows because $f^*(e) > 0$, which means that e is part of some flow path in f^*, and therefore its delay is at most that of the whole path. ■

Note: If there is an $s - t$ path of good edges, then $D \leq (n-1)D^*$.

Looking at the example in the figure, we see that in this case only the edges sv and ut are good. So not every graph contains such a path of good edges. However, we will use something similar in order to establish the theorem.

Claim 3 *Nash flow is acyclic (i.e., edges with $f(e) > 0$ do not form cycles.*
Proof sketch: Nash flow is optimal for another objective function ($c_e(x) = \int_0^x d_e(y)dy$), and decreasing flow around a cycle decreases this cost function. ■

Suppose that we sort the nodes $s, v_1, v_2, ..., t$ in the order of delay on the path from s to v, so that $d(v_i) \leq d(v_{i+1})$. Notice that if $f(ij) > 0$, then $d(i) \leq d(j)$. As a result, since the graph of edges that carry flow f is acyclic, all edges that carry flow will go from earlier nodes to later nodes in this ordering. Our example would become:

$$\begin{tikzcd} \boxed{s} \arrow[r] & \boxed{u} \arrow[r] & \boxed{v} \arrow[r] & \boxed{t} \end{tikzcd}$$

(the edges shown are the ones carrying flow f).

Claim 4 *$d(v_{i+1}) \leq d(v_i) + D^*$ for all i.*
Notice that this implies the claimed bound of $D \leq (n-1)D^*$.
Proof. Let $S = \{s, v_1, v_2, ..., v_i\}$ be some prefix of the ordering. We will prove that there exists a good edge uv from S to $\bar{S}$. This suffices to prove the claim because $d(v_{i+1}) - d(v_i) \leq d(v) - d(u) \leq D^*$

by Claim 2. If r is the total $s-t$ flow, then, using the fact that edges with $f(e) > 0$ all go from left to right, we know that the flow f out of S is exactly r. The flow f^* out of S is at least as much, even if there are paths that reenter S. So there must be some edge for which $f^*(uv) \geq f(uv)$, which proves the claim. ■

We have shown that, although there may not be a path of good edges from s to t, there is a good edge crossing every cut in the ordered list of nodes. In the example, these are edges sv and ut.

2 Bad example for two commodities

The following figure shows that if the graph contains two source-sink pairs, then the ratio of Braess' can be exponential in n.

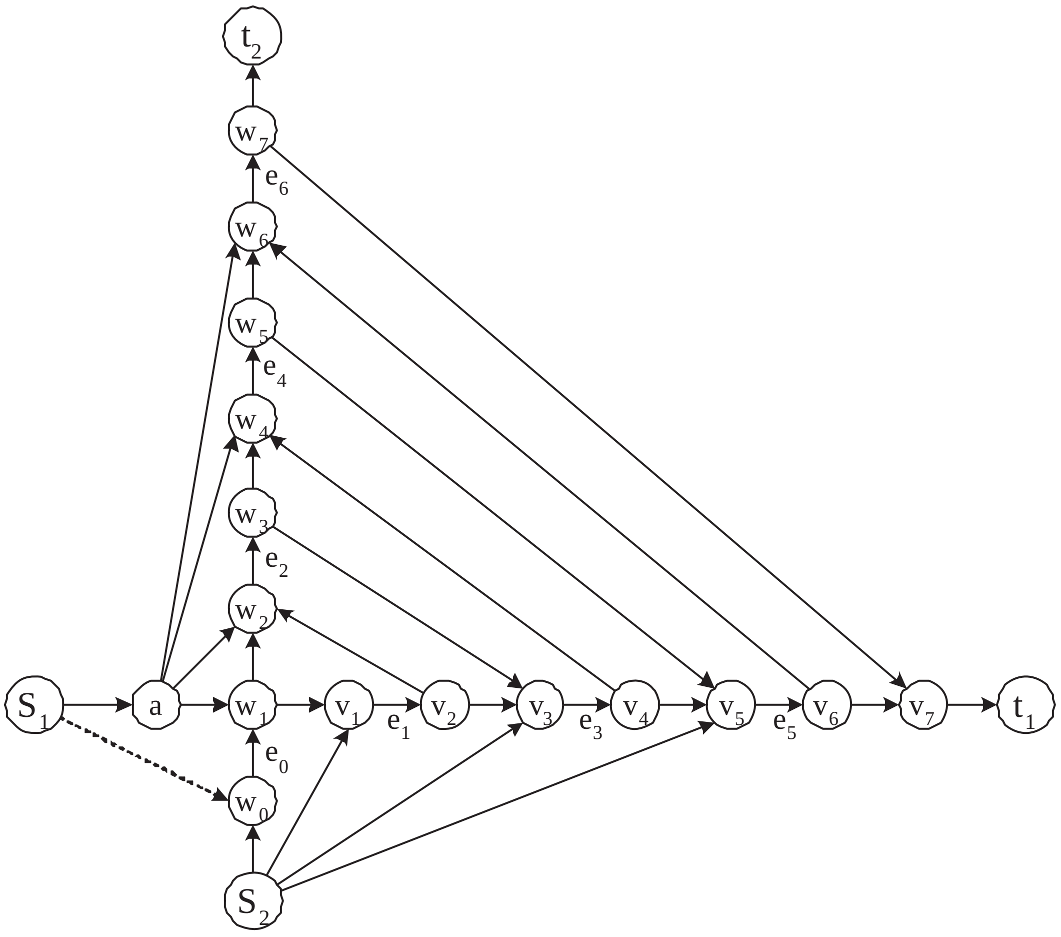


Figure 1: The users taking diagonal "shortcuts" results in delays that grow as a Fibonacci sequence.

CS 684 Algorithmic Game Theory	**Scribe: Ara Hayrapetyan**
Instructor: Eva Tardos	**May 7, 2004**

The final problem that we will study during this last lecture is the stable matching problem viewed from a game theoretic standpoint. The stable matching problem arises in many practical settings the most prominent of which is the assignment of hospital residencies. Each candidate knows his preferred hospitals and each hospital knows the candidates it prefers. The goal is to find a "good" assignment of candidates to hospitals, where "good" will be defined shortly. We will start by overviewing the problem and the well known proof of existence of stable matchings. Then we will shift to more game theoretic aspects of the problem.

1 The Stable Matching Problem

In this problem we are given n men and n women and each person has a full perference list of the people of opposite gender. Our goal is to find a pairing (matching) of men and women that in some sense is stable. More formally we say a matching is *unstable* if there exists a pair of couples $(m_1, w_1), (m_2, w_2)$ such that m_1 and w_2 perfer each other to their current partners. A matching is stable if there are no unstable couples. Our goal once again is to find stable matchings.

There is a well known algorithm for finding stable matchings. The algorithm consists of men proposing women in order of preference and women holding on to their best proposal so far (obviously the symmetric algorithm of women proposing also works). It is well known that this algorithm finds stable matchings and the the final matching doesn't depend on the order of proposals (notice that the algorithm above was underspecified in that regard). Also all men are paired with their best stable partner in the outcome of this algorithm and all women are paired with their worst stable partner.

2 The Game Theory Question

Is this mechanism truthful? Is it truthful for men? Is it truthful for women? In the rest of this lecture we will address these questions in a couple of different settings.

First, it can be shown (though it is nontrivial) that the above mechanism is truthful for men. However, it is not truthful for women. To anaylize the truthfulness of the above mechanism with respect to women we will consider a more restricted setting where men have a partial preference list of at most k women and if none of those women prefer them then they'd rather stay single. Women on the other hand do have the full preferece list of men. The motivation of this is that men often indeed can't rank all the women in the world, yet women never need to have a preference list - all they need to be able to do is to decide between two alternatives, which they surely can.

Again it can be shown that the mechanism produces a stable matching, where the notion of instability has to be modified to include running off with a single person. Similarly it can be shown that again the order of proposals is insignificant, and that men are paired with their best partners in any stable matching and women are paired with their worst partners (again we have to adjust best and worst to account for single people).

For this model as we will see with a small example the algorithm above is not truthful for women (it can be shown to be truthful for men). There is in fact a theorem stating that there is no truthful mechanism for this game.

The small example is the following - there are three men m_1, m_2, m_3 and three women w_1, w_2, w_3. and the preference list for w_2 is (m_3, m_1, m_2). Let us assume that the current couples are (m_i, w_i), w_2 is currently being offered by m_1, and w_3 prefers m_1 to all other men. If w_2 forgoes the offer by m_1 and stays with m_2 then m_1 will offer to w_3 which will open m_3 who will offer to w_2 thus w_2 will get his best choice. Therefore the mechanism isn't truthful.

But not all hope is lost. We can still show some positive things. In parcitular we will consider the case when all men choose $\leq k$ women in their preference lists and do so randomly. We will show (outline the proof) that in this scenario the number of women who would benefit from lying depends only on k and not on the number of total people.

Theorem 1 *Consider the problem instance where we are given a full preference lists of women and men choose at most k women at random in their preference list. Then the expected number of women that would benefit from lying is no more than $O(e^k)$.*

Proof: The proof will follow from a sequence of observations.

1. A woman can only benefit from a lie if she is in multiple stable matchings. Indeed if the women lies and benefits the resulting matching has to also stable. So now we will instead evaluate the probability that a woman is in more than one stable matching.

2. Women can defer their lie until they arrive at a stable matching, at which point they can discard their current partner. This will create a cascade effect (by running the stable matching algorithm again) and for her to be better off she has to get at least one other offer (since being single is the worst outcome for women). So we now only have to evaluate the probability that the woman will get a second offer after discarding her partner at the end of the run of the matching algorithm.

3. Let us analyze the run of the algorithm once a certain woman w discards his partner. What will happen is that the free man will make an offer, which will either be rejected or a new man will be freed. The process can stop in three ways.

   - w gets an offer (this is the good scenario for w)
   - The current available man has no more offers and chooses to be single.
   - A single woman gets an offer.

   It is sufficient for us to prove that the last event in the above listing is a high probability event relative to the first one, which would imply that w doesn't have much of a chance of getting another offer. So for now ignore the second event. The probability that w will be offered before any single women is $\frac{1}{|S|+1}$ where S is the set of single women (recall that the lists are randomly generated). On the other hand the probability that a woman is single is at least the probability that she is on nobody's list, and since the lists are generated randomly the latter is at most

$$(1 - \frac{1}{n})^{nk} \approx e^{-k}. \tag{1}$$

And therefore the expected number of single women is at least ne^{-k}. So therefore the probability that w benefits from lying is no more than the probability that she gets a second offer, which in turn is at most $\frac{1}{ne^{-k}}$. Thus the expected number of women that could benefit from lying is at most $\frac{n}{ne^{-k}} = e^k$ - a quantity that depends only on k. Thus the proof is complete.