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.