Computational Game Theory **Spring Semester, 2011/2012**

Lecture 2: March 14

Lecturer: Amos Fiat *Scribe: Hadas Zur & Alon Ardenboim*

2.1 Introduction

Informally, a set of strategies is a *Nash equilibrium* if no player can do better by unilaterally changing his or her strategy. To see what this means, imagine that each player is told the strategies of the others. Suppose then that each player asks himself or herself: "Knowing the strategies of the other players, and treating the strategies of the other players as set in stone, can I benefit by changing my strategy?" If any player would answer "Yes", then that set of strategies is not a Nash equilibrium. But if every player prefers not to switch (or is indifferent between switching and not) then the set of strategies is a Nash equilibrium. Thus, each strategy in a Nash equilibrium is a best response to all other strategies in that equilibrium. The Nash equilibrium may sometimes appear non-rational in a third-person perspective. This is because it may happen that a Nash equilibrium is not Pareto optimal. The Nash equilibrium may also have non-rational consequences in sequential games because players may "threaten" each other with non-rational moves. For such games the subgame perfect Nash equilibrium may be more meaningful as a tool of analysis.

Thus the outcome of rational behavior might be inefficient. Nash equilibria in general and user equilibria in particular are typically inefficient: they generally do not minimize the social cost. An example of an inefficient equilibrium can be seen in the prisoner's dilemma (see Fig. 2.1). The only equilibrium in this game, when played by rational players, is that they both are tattletales, while it is clear that if they both keep quiet, the outcome is favorable for them both. Koutsoupias and Papadimitriou (1999) proposed to analyze the inefficiency of equilibria from a worst-case perspective; this led to the notion of "price of anarchy" (Papadimitriou 2001), which is the ratio of the worst social cost of a Nash equilibrium to the cost of an optimal solution. In the context of selfish routing (i.e, the traffic model which will be described later), the price of anarchy was analyzed in a series of papers for increasingly more general classes of cost functions and model features; see, among others, Roughgarden and Tardos (2002), Roughgarden (2003), Schulz and Stier-Moses (2003), Chau and Sim (2003), Correa, Schulz, and Stier-Moses (2004), Roughgarden and Tardos (2004), Perakis (2004), and Roughgarden (2005). [CorreaJ05]

In order to measure the inefficiency of an equilibrium, we'll introduce the concepts of *Price of Anarchy* (**PoA**), *Price of Stability* (**PoS**) and *Strong Price of Anarchy* (**SPoA**),

P1/P2	Tattletale	Keep quiet
Tattletale	3,3	0,5
Keep quiet	5,0	1,1

Figure 2.1: The prisoner's dilemma.

and show how to measure them in a variety of games such as selfish routing and load balancing problems. One should mention there are other concepts to be measured as well.

The efficiency is measured in the context of the goal function decided upon. For example, let u_i denote player i's utility, then we could look at a utilitarian goal function such as $\sum_i u_i$, that is, maximizing the *social welfare* (*utilitarianism* is an ethical theory holding that the proper course of action is the one that maximizes the overall "happiness"). On the other hand, a more egalitarian goal function, such as to maximize $\min_i u_i$ could be desired (*egalitarianism* is a trend of thought that favors equality of some sort among living entities).

2.1.1 Common Measures

In order to measure the inefficiency of games, we need to specify:

- The objective function in use.
- A definition of an approximately optimal solution.
- A definition of an equilibrium.
- If more than one equilibrium exists, which one do we consider.

Our focus is mainly on the equilibria depicted in figure 2.2. We'll give a quick recap of some of them for sake of clarity:

1. **Pure Nash Equilibrium** is an equilibrium achieved when players are using pure strategies. A pure strategy provides a complete definition of how a player will play a game. In particular, it determines the move a player will make for any situation he or she could face. A player's strategy set is the set of pure strategies available to that player. The Prisoners dilemma is a pure Nash game.
2. **Mixed Nash Equilibrium** is an equilibrium achieved when players are using mixed strategies (as described in lecture 1). A mixed strategy is an assignment of a probability to each pure strategy. This allows for a player to randomly select a pure strategy. Since

probabilities are continuous, there are infinitely many mixed strategies available to a player, even if their strategy set is finite.

3. **Strong Nash Equilibrium** A strong Nash equilibrium is a Nash equilibrium in which no coalition, taking the actions of its complements as given, can cooperatively deviate in a way that benefits all of its members. While the Nash concept of stability defines equilibrium only in terms of unilateral deviations, strong Nash equilibrium allows for deviations by every conceivable coalition. This equilibrium concept is particularly useful in areas such as the study of voting systems, in which there are typically many more players than possible outcomes, and so plain Nash equilibria are far too abundant.

4. **Correlated Equilibrium** A correlated equilibrium is a solution concept that is more general than the well known Nash equilibrium. It was first discussed by mathematician Robert Aumann (1974). The idea is that each player chooses his/her action according to his/her observation of the value of the same public signal. A strategy assigns an action to every possible observation a player can make. If no player would want to deviate from the recommended strategy (assuming the others don't deviate), the distribution is called a correlated equilibrium.

5. **Best Response Dynamic** is a dynamic in which each player chooses to act in a "best response" way to the situation at hand. Such a dynamic can converge, but this is not always the case.

6. **No Regret Dynamic** is a dynamic in which each player takes a "best response" action according to a distribution on past events. It can be shown that such a dynamic converges into a mixed Nash equilibrium.

Our main inefficiency concepts are:

Definition *Price of Anarchy* is the result of $\frac{\text{Worst NE}}{\text{OPT}}$. The price of anarchy (PoA) is a concept in game theory that measures how the efficiency of a system degrades due to selfish behavior of its agents. It is a general notion that can be extended to diverse systems and notions of efficiency. For example, consider the system of transportation of a city and many agents trying to go from some initial location to a destination. Let efficiency in this case mean the average time for an agent to reach the destination. In the "centralized" solution, a central authority can tell each agent which path to take in order to minimize the average travel time. In the "decentralized" version, each agent chooses its own path. The Price of Anarchy measures the ratio between average travel time in the two cases.

Usually the system is modeled as a game and the efficiency is some function of the outcomes (e.g. maximum delay in a network, congestion in a transportation system, social welfare in an auction, ...). Different concepts of equilibrium can be used to model the selfish

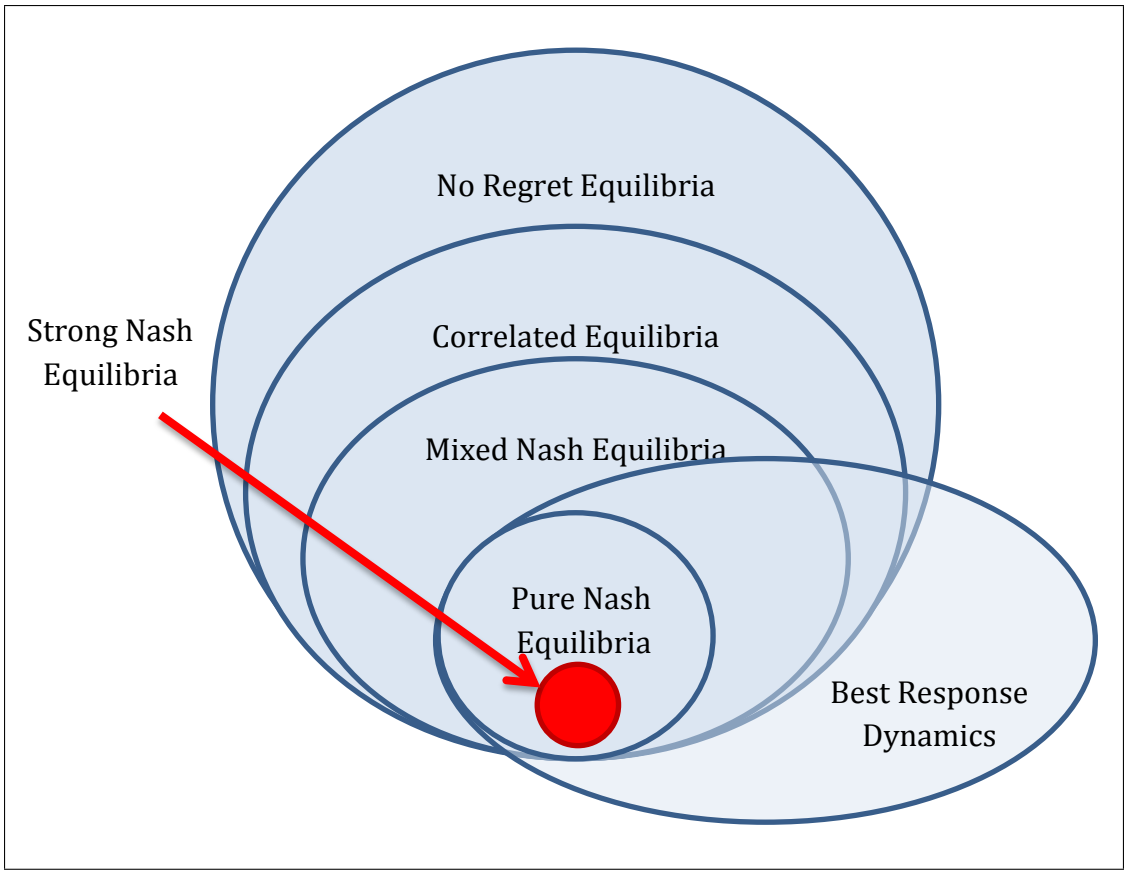


Figure 2.2: Equilibrium concepts.

behavior of the agents, among which the most common is the Nash Equilibrium. Different flavors of Nash Equilibrium lead to variations of the notion of Price of Anarchy as Pure Price of Anarchy (for deterministic equilibria), Mixed Price of Anarchy (for randomized equilibria), Bayes-Nash Price of Anarchy (for games with incomplete information), ... Other notions, other than Nash, lead to variations of the concept, as the Price of Sinking . The term Price of Anarchy was first used by Koutsoupias and Papadimitriou [CorreaJ05] but the idea of measuring inefficiency of equilibrium is older. The concept in its current form was designed to be the analogue of the "approximation ratio" in an Approximation algorithm or the "competitive ratio" in an Online algorithm.

The term Price of Anarchy was first used by Koutsoupias and Papadimitriou [CorreaJ05] but the idea of measuring inefficiency of equilibrium is older. The concept in its current form was designed to be the analogue of the 'approximation ratio' in an Approximation algorithm or the 'competitive ratio' in an Online algorithm.

Definition *Price of Stability* is the results of $\frac{\text{Best NE}}{\text{OPT}}$. The price of stability can be viewed as how good could it be if we could "intervene" in the game until a certain point.

One can look at the price of anarchy as how much we can loose if we let players act as they will, while the price of stability can be viewed as how good could it be if we could "intervene" in the game until a certain point. PoA is similar to other concepts used in computer science to measure inefficiency (approximation ratio, competitive ratio, etc.).

In accordance to the many kinds of equilibria concepts depicted above, we can talk about PoA and PoS in the context of strong\pure\mixed equilibria etc., and get strong\pure\mixed PoA\PoS. Since the more reduced the set of equilibria is, the best and worst equilibria are less good and bad respectively, we can see that for maximization problems (when PoA and PoS ≤ 1)

$$\text{Correlated PoA} \leq \text{Mixed PoA} \leq \text{Pure PoA} \leq \text{Strong PoA} \tag{2.1}$$

and, on the other hand

$$\text{Correlated PoS} \geq \text{Mixed PoS} \geq \text{Pure PoS} \geq \text{Strong PoS} \tag{2.2}$$

, where pure and strong PoA\PoS don't always exist (the inequalities are in the opposite direction when we talk about minimization problems).

2.2 Finding the PoA and PoS in Various Games

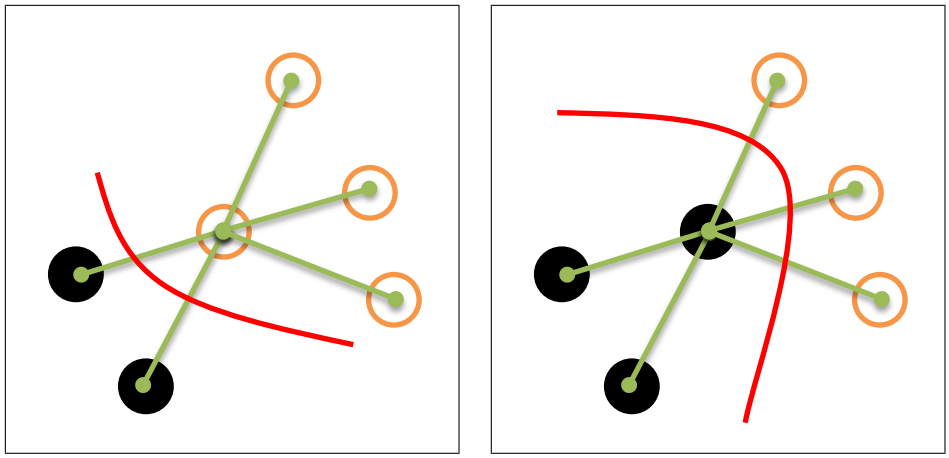
To demonstrate the concepts above, we'll take the prisoner's dilemma with the values depicted in figure 2.1. We'll take a social cost goal function (we want to minimize the overall number of years in prison). Since the only equilibrium in this game is where both players tattle on each other, both the worst and best (pure) Nash give us a cost of $3+3=6$ years in prison, where the optimal action costs 2 years in prison. Therefore, both the PoA and PoS in this case are 3. We can see that we can change the values of table in figure 2.1 and get PoA and PoS values as big as we will. Thus, for this game, the PoA and PoS are unbounded. In this section, we'll show games where we can bound the PoA and PoS, and demonstrate methods for doing so.

2.2.1 Max-cut Game (Arranging Kids on the Playground)

For a graph, a maximum cut is a cut whose size is at least the size of any other cut. The problem of finding a maximum cut in a graph is known as the max-cut problem. The problem can be stated simply as follows. One wants a subset S of the vertex set such that the number of edges between S and the complementary subset is as large as possible. There is a more advanced version of the problem called weighted max-cut. In this version each edge has a real number, its weight, and the objective is to maximize not the number of edges but the total weight of the edges between S and its complement. The weighted max-cut problem is often, but not always, restricted to non-negative weights, because negative weights can change the nature of the problem.

The following decision problem related to maximum cuts has been studied widely in theoretical computer science: Given a graph G and an integer k, determine whether there is a cut of size at least k in G. This problem is known to be NP-complete. It is easy to see that problem is in NP: a yes answer is easy to prove by presenting a large enough cut. The NP-completeness of the problem can be shown, for example, by a transformation from maximum 2-satisfiability (a restriction of the maximum satisfiability problem). The weighted version of the decision problem was one of Karp's 21 NP-complete problems; Karp showed the NP-completeness by a reduction from the partition problem. The canonical optimization variant of the above decision problem is usually known as the maximum cut problem or max-cut problem and is defined as: Given a graph G, find a maximum cut.

Approximation algorithms: There is a simple randomized 0.5-approximation algorithm - for each vertex flip a coin to decide to which half of the partition to assign it. In expectation, half of the edges are cut edges. This algorithm can be derandomized with the method of conditional probabilities; therefore there is a simple deterministic polynomial-time 0.5-approximation algorithm as well. One such algorithm is: given a graph $G = (V, E)$ start



(a) A node before improving its state. (b) A node after improving it's state.

Figure 2.3: An illustration of the cut in a scenario where a node would benefit by switching sides. Figure (a) shows the node in the "white" group with more neighbors on its side. Figure (b) shows the node after switching sides. As a result, the cut size increases (the cut is illustrated by the red line).

with an arbitrary partition of V and move a vertex from one side to the other if it improves the solution until no such vertex exists. The number of iterations is bound by $O(|E|)$ because the algorithm improves the cut value by at least 1 at each step and the maximum cut is at most $|E|$. When the algorithm terminates, each vertex $v \in V$ has at least half its edges in the cut (otherwise moving v to the other subset improves the solution). Therefore the cut is at least $0.5|E|$. The best known max-cut algorithm is the 0.878-approximation algorithm by Goemans and Williamson using semidefinite programming and randomized rounding. It has been shown by Khot et al that this is the best possible approximation ratio for Max-Cut assuming the unique games conjecture.

The max-cut game: Let's consider the following game - we are given a graph $G = (V, E)$ where the players are the nodes $v \in V$. Imagine the players being kids on the playground. An edge $(u, v) \in E$ means u and v "hate" each other. Each kid has to choose a side on the playground (chose a strategy between the "black" and "white"). The utility of node v is the number of neighbors of different color. We'll consider a social welfare function as our goal function.

Claim 2.1. *A max-cut, where the nodes in each side of the cut are the same color, defines a NE state in the max-cut game (hence the name).*

Proof. Let's consider a max cut state as depicted in figure 2.3. Let's assume there exists a node that would improve its position by moving to the other side. The only way this is

an improvement for it is if it has more neighbors on its side than on the other side, but then by moving it to the other side we'll increase the size of the cut, in contradiction to the maximality of the cut. □

One can point out that the maximal cut is also the state that maximizes our social function, since the social welfare is equivalent to the number of edges in the cut. We get that the PoS in this case is 1 which seems promising. But there isn't any reason to assume that the equilibrium received in such a game will be the max-cut, since finding the max-cut is computationally hard (and we assume the market "operates" in polynomial time). Given an instance of the max-cut game, here's a polytime algorithm which gives us a NE that approximates the optimal NE for this game:

Algorithm 1 Greedy-Find-Cut (GFC)

Require: $G = (V, E)$.
Ensure: An equilibrium with 2-approximation to the optimal social welfare.

1: Let B, W be an arbitrary division of the nodes into two sets.
2: For $v \in V$, let $B[v]$ denote the set of neighbors of v in B ($W[v]$ is defined equivalently).
3: **while** $(\exists v \in B.B[v] > W[v]) \bigvee (\exists v \in W.W[v] > B[v])$ **do**
4: v switches sides (if v in B, move it to W and vice-versa).
5: **end while**

Claim 2.2. ***Algorithm 1*** *runs in polynomial time and gives a 2-approximation to the optimal social welfare. Furthermore, at the end of the algorithm we reach a NE state.*

Proof. Since in each iteration, the cut is increased by at least one edge, and the size of the cut cannot be bigger than $|E|$, the number of iterations is bounded by $|E|$. To derive the approximation factor, we notice that for each vertex the number of edges of it crossing the cut is $\geq$ than the number of edges staying on the same side. Therefore, the size of the cut is at least $\frac{|E|}{2}$, while the optimal cut is obviously smaller than $|E|$ in size.
Let's assume that the state at the end of the algorithm is not an equilibrium state. Then there's at least on vertex which benefits from switching sides, but this can happen only if it has more neighbors on the other side, which means that the algorithm won't stop in this case. □

In addition, it is easy to see that the at the wort possible equilibrium, the social welfare is at least half of the optimal (hint: what can we say about the number of neighbors of a given vertex on his side and on the other side at an equilibrium). We get a bounded PoA of $\frac{1}{2}$ in this game.

2.2.2 Traffic Flow and Wardrop Equilibrium

Congestion games (Rosenthal 1973) are non-cooperative games in which a player's strategy is to choose a subset of resources, and the utility of each player only depends on the number of players choosing the same or some overlapping strategy. The most prominent example of a nonatomic congestion game is the traffic routing model of Wardrop (1952). The arcs in a given network represent the resources, the different origin-destination pairs correspond to the player types, and the strategies available to a particular player type are the paths in the network between its origin-destination pair. The cost of an arc describes the delay experienced by traffic traversing that arc as a function of the flow on that arc. A social optimum corresponds to a multicommodity flow of minimum total delay, while a Nash equilibrium equals a user equilibrium flow, where every player is traveling on a shortest path under the prevailing conditions. [CorreaJ05]

Wardrop equilibria are commonly used as a solution concept of network games when modeling transportation and telecommunication networks with congestion. This concept assumes that players select a route that minimizes the time or cost incurred in its traversal. This behavioral assumption admits convenient mathematical descriptions, and efficient algorithms for the computation of equilibria are available. For that reason, planners have been making use of this concept for decades for evaluating projects, optimizing tolls, estimating demands, and a myriad of applications arising from extensions of the basic model. [CorreaJR10]

A common behavioral assumption in the study of transportation and telecommunication networks is that travelers or packets, respectively, choose routes that they perceive as being the shortest under the prevailing traffic conditions. The situation resulting from these individual decisions is one in which drivers cannot reduce their journey times by unilaterally choosing another route, which prompted Knight to call the resulting traffic pattern an equilibrium. Nowadays it is indeed known as the Wardrop (or user) equilibrium, and it is effectively thought of as a steady-state evolving after a transient phase in which travelers successively adjust their route choices until a situation with stable route travel costs and route flows has been reached. In a seminal contribution, Wardrop stated two principles that formalize this notion of equilibrium and the alternative postulate of the minimization of the total travel costs. His first principle reads: The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route. Wardrop's first principle of route choice, which is identical to the notion postulated by Kohl and Knight, became accepted as a sound and simple behavioral principle to describe the spreading of trips over alternate routes due to congested conditions. Since its introduction in the context of transportation networks in 1952 and its mathematical formalization by Beckmann, McGuire, and Winsten, transportation planners have been using Wardrop

equilibrium models to predict commuters decisions in real life networks. These models have been and are still used today to evaluate alternative future scenarios and decide a route of actions. [CorreaJR10]

We formulate Wardrop equilibria using the following mathematical model:

- The network topology is given in the form of a directed graph $G = (V, E)$.
- There are k pairs of source and destination vertices:

$$(s_1, t_1), \ldots, (s_k, t_k).$$

- Each source-destination pair has an associated amount of data r_i (rate) to transfer from s_i to t_i.
- Each edge $e \in E$ has an associated cost function $c_e(\cdot)$ (sometimes regarded as a latency function).
- The cost of the flow is the sum of the costs along the paths of the flow:

$$c(f) = \sum_{e \in f} f_e \times c_e(f_e) \tag{2.3}$$

  where f_e is the amount of flow transferred through edge e.

- The goal function is naturally to minimize the cost of the flow (social welfare function).

In the following examples, we'll regard the flow as non-atomic, meaning that the flow can be divided into infinitely small fractions and to be transferred via multiple paths. In the next section, we'll consider a special category of routing problems in which the flow is atomic, that is, cannot be divided to different paths.

We can view each fraction of a flow as a selfish player routing itself in a best response manner to the network at hand with the goal of using the route of minimal cost. The equilibrium received, in this case, is the one where each fraction would use a route of bigger cost, if forced to use a different route.

Take for example the network depicted in figure 2.4 with two edges and only one source-destination pair and an amount of size 1 to transfer. Let the first cost function be $c_2(x) = 1$ representing an "infinitely" wide edge, with no dependence on the amount of traffic and the second cost function be $c_1(x) = x$ where there is a one-to-one correspondence between the amount of traffic on the edge and the time it takes to pass it. It can be derived using

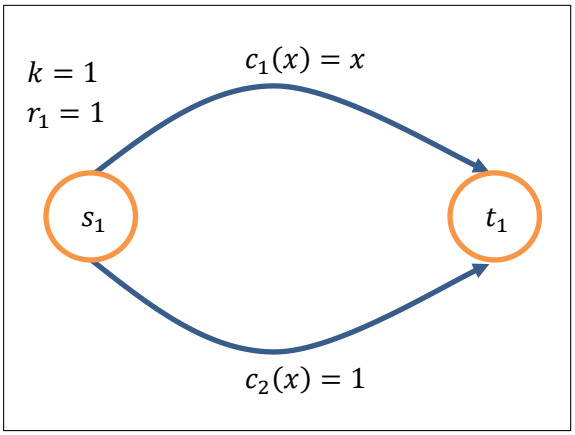


Figure 2.4: Example of a network with an inefficient Waldrop equilibrium.

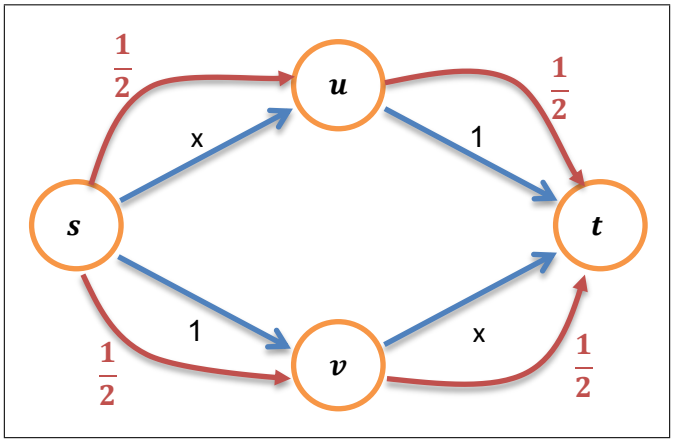
first order conditions that the optimal routing in this case transfers half of the amount using the upper edge, and half through the lower edge. This incurs in a cost of $\frac{1}{2}\cdot c_1(\frac{1}{2})+\frac{1}{2}\cdot c_2(\frac{1}{2}) = \frac{3}{4}$.

On the other hand, the only equilibrium in this case is the one that passes the entire flow through the upper edge. It is clear that each fraction in the flow of the upper edge would not improve its route when using the lower edge. On the other hand, each fraction in the lower flow would improve its route when moving to the upper edge. Therefore, the cost in an equilibrium is 1, and the PoA (and PoS) is $\frac{4}{3}$.

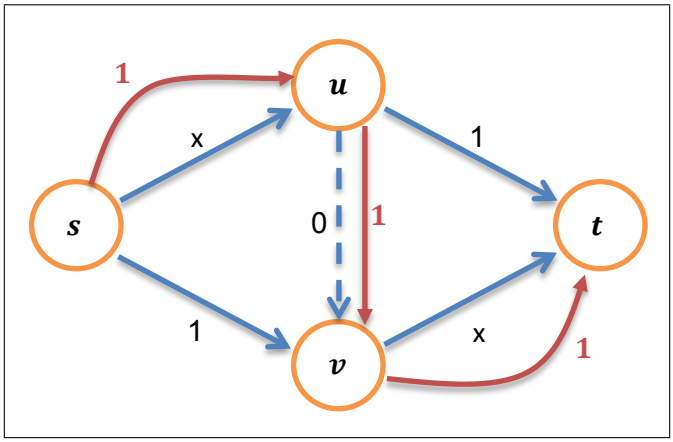
An interesting phenomenon in the context of network planning is that adding more paths to the network can result in a less efficient data transfer. This can be shown in the famous Braess's paradox. Braess's paradox, credited to the mathematician Dietrich Braess, states that adding extra capacity to a network when the moving entities selfishly choose their route, can in some cases reduce overall performance. This is because the Nash equilibrium of such a system is not necessarily optimal.

Take for example the network depicted in figure 2.5(a). The single equilibrium in this case is the one that passes half of the flow using the lower route and half of it using the upper one. The cost of this equilibrium is $(\frac{1}{2} + \frac{1}{4}) \cdot 2 = 1.5$. If we add another edge from u to v with cost function $c(x) = 0$, we now get that the new equilibrium passes all the flow to u, takes the edge (u, v) with cost 0, and then uses (v, t). The cost of this equilibrium is 2. We get a situation that seems absurd at first - we get a less efficient outcome even though there are more resources available.

If the latency functions are linear then adding an edge can never make total travel time

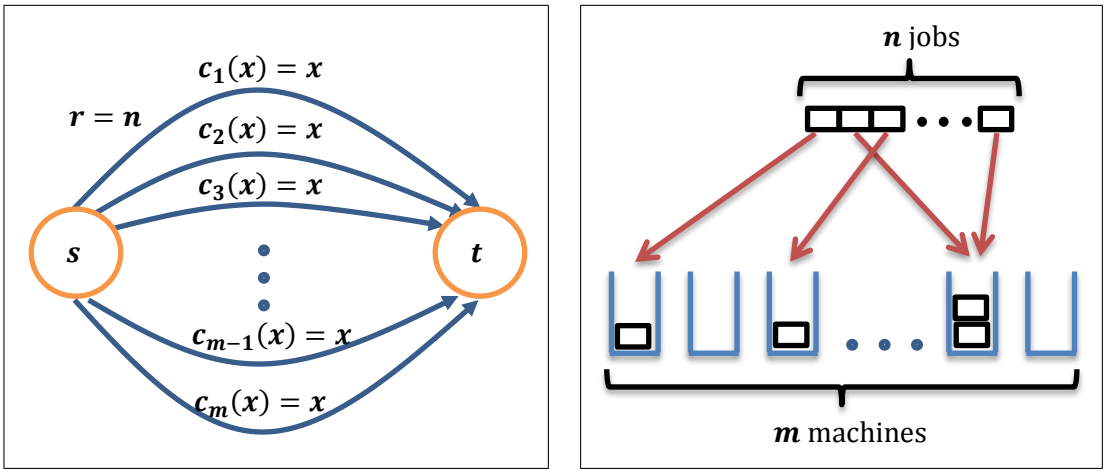


(a) The network before the adding an edge.



(b) The network after adding the edge (u, v) (marked by the dotted blue line).

Figure 2.5: Braess paradox. The red arrows mark the flow in the network. Adding an edge to network, as shown in figure (b), only increases the cost of the flow.



(a) Load balancing presented in its network form. (b) Load balancing presented using job scheduling on machines.

Figure 2.6: The two equivalent forms of presenting the load balancing game.

at equilibrium worse by a factor of more than $\frac{4}{3}$. [EaselyD10]

2.2.3 Load Balancing

Consider the following special and interesting form of routing game (see Fig. 2.6) in which the flow is atomic, and there are m parallel links to transfer n units of communication request. An analogous way to view these settings is the load balancing framework in which m machines are performing n tasks. The cost of each link corresponds to the load on a machine. Our interest in these settings is to minimize the maximum load on a machine (equivalent to minimizing the maximal delay on a link).
First, we set our focus on the identical machine in which it takes each machine the same time to process the same job. Let's consider the case in which all the jobs have the same processing time for warmup:

Claim 2.3. *The pure PoA for the identical machine load-balancing game in which the jobs have an identical weight is 1.*

Proof. Let z be the load on the maximal machine at an equilibrium. WLOG, consider the machines to be ordered from the most loaded machine to the least loaded machine (see Fig. 2.7). The only pure equilibrium is the one in which we first have machines of load z, and afterwards (if $m \cdot z \neq n$) we have machines of load $z-1$. No job gains by being processed on a different machine, and if we consider a state where there exists a machine of load $< z-1$ then there are jobs that can gain by transferring to that machine. It's also clear that the optimum must look as in figure 2.7. □

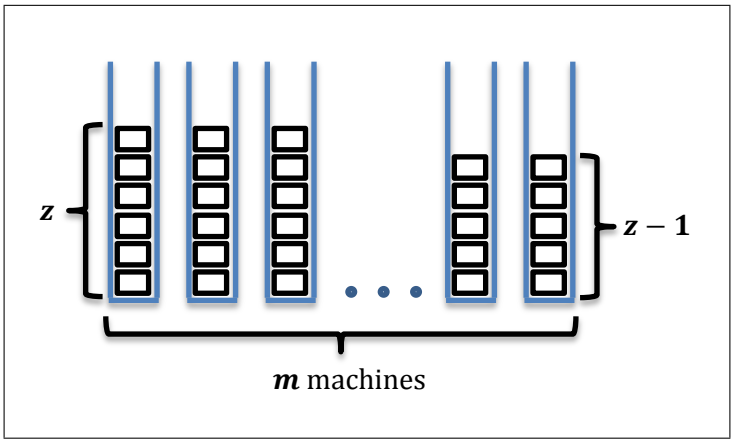


Figure 2.7: The equilibrium in the identical machine load balancing game where the jobs have identical wights. The machined are ordered by wight. This is also the optimum.

For the case where the jobs have an arbitrary processing time our PoA is not as good, but still bounded by a constant:

Claim 2.4. *The pure PoA for the identical machine load-balancing game in which the jobs have arbitrary weights is bounded by* $2 - \frac{2}{m+1}$.

Proof. Let OPT denote the maximal load in the optimal division of jobs to machines. Let H_L denote the highest load of any machine in an equilibrium. To prove the bound, we notice the following for an equilibrium:

- In case $H_L >$ OPT, there must be at least two jobs on the most loaded machine. Therefore, the lightest job on the most loaded machine weighs at most $\frac{H_L}{2}$.
- Furthermore, every machine other than the most loaded must have a load of at least $\frac{H_L}{2}$, otherwise, the lightest job on the machine with the highest load would prefer to be executed on a different machine.
- Since the optimum is bigger than any average of loads on machines, we get

$$\begin{aligned} \text{OPT} &\geq \frac{H_L + (m-1) \cdot H_L/2}{m} && (2.4) \\ &= H_L \cdot \left(\frac{m+1}{2m} \right). && (2.5) \end{aligned}$$

rearranging equation 2.4 gives us

$$\text{PoA} = \frac{H_L}{\text{OPT}} \leq 2 - \frac{2}{m+1}. \quad (2.6)$$

Variant	Pure PoA	Mixed PoA
m identical machines, identical jobs	1	$\Theta\left(\frac{\log m}{\log\log m}\right)$
m identical machines	$2-\frac{2}{m+1}$	$\Theta\left(\frac{\log m}{\log\log m}\right)$
2 related machines	1.618	-
m related machines	$\Theta\left(\frac{\log m}{\log\log m}\right)$	$\Theta\left(\frac{\log m}{\log\log\log m}\right)$
m machines restricted assignment	$\Theta\left(\frac{\log m}{\log\log m}\right)$	$\Theta\left(\frac{\log m}{\log\log\log m}\right)$

Figure 2.8: A summary of price of anarchy results regarding variant of the load balancing game.

□

As stated in formula 2.1, considering more equilibria can result in a worse PoA value. For the case of identical machine and identical jobs, we have the following:

Claim 2.5 (unproved)**.** *The mixed price of anarchy for the identical machine load-balancing game is* $\Theta\left(\frac{\log m}{\log\log m}\right)$.

Other interesting variants of the problems are related machines and restricted assignment. In the case of *related machines* each machine i has a different processing speed v_i, and the load on the machine is calculated using the following formula:

$$L_i = \frac{1}{v_i} \cdot \sum_{j|A(j)=i} w_j \tag{2.7}$$

where $A(j)$ denotes the machine job j was assigned to. *Restricted assignment* gives each job a subset of machines it can be assigned to. Table 2.8 sums up some major results for variants of the load balancing game.

We'll prove the lower bound on the PoA for the case of related machines (the proof of the upper bound will be given in the notes of the next lecture).

Claim 2.6. *The lower bound of the pure PoA for the load balancing game with m related machines is* $\omega\left(\frac{\log m}{\log\log m}\right)$.

Proof. Consider the situation depicted in figure 2.9 where there are $k+1$ groups of machines $G_1, \ldots, G_i, \ldots, G_k$. We'll now describe the groups:

- G_0 contains 1 machine with velocity $v = 2^k$ which executes k jobs weighing $w = 2^k$ each.

- G_1 contains k machines with velocity $v = 2^{k-1}$. Each machine executes $k-1$ jobs weighing $w = 2^{k-1}$ each.

- G_2 contains $k \cdot (k-1)$ machines with velocity $v = 2^{k-2}$. Each machine executes $k-2$ jobs weighing $w = 2^{k-2}$ each.

- Generally, G_i contains $\frac{k!}{(k-i)!}$ machines with velocity $v = 2^{k-i}$. Each machine executes $k-i$ jobs weighing $w = 2^{k-i}$ each

- G_{k-1} contains $k!$ machines with velocity $v = 2$. Each machine contains 1 job weighing $w = 2$.

- G_k contains $k!$ machines with velocity $v = 1$. Each machine is free.

First we'll show this is an equilibrium. Using equation 2.7 we can deduce that the load on the machines in G_i is $k-i$. Let's assume there exists a job that wants to move to a machine in G_i. Each job on a machine in $G_{i'}$ where $i' \geq i$ would surely move to a machine with a bigger load, so let's consider only machines in $G_{i'}$ where $i' < i$. Moving the job from machine in i' to a machine in i would increase its load in $\frac{2^{k-i'}}{2^{k-i}} = 2^{i-i'} > i - i'$. Therefore, after moving the job, the load on the new machine is strictly bigger than $k - i + i - i' = k - i'$, that is, larger than the load the job arrived from. The most loaded machine in this equilibrium has a load of k. Since $m \sim k!$, we have that $k \sim \frac{\log m}{\log \log m}$.

Let's consider the following iterative process - take all the $k!$ jobs on machines from G_{k-1} and move them to the machines in G_k so that each machine gets one job. Afterwards, take the $k!$ jobs on machines from G_{k-2} (there are $\frac{k!}{2}$ machines, each processes 2 jobs) and move them to the machines in G_{k-1} so that each machine get's one job. Generally, take the $\frac{k! \cdot (k-i)}{(k-i)!} = \frac{k!}{(k-i-1)!}$ jobs from the machines in G_i and move them to the $\frac{k!}{(k-i-1)!}$ machines in G_{i+1} so that each machine gets one jobs. Continue until all the k jobs from G_0 move to the k machines in G_1. After moving the jobs "upwards", we notice that each machine that has a velocity of $v = 2^{k-i}$ processes a single job with weight $w = 2^{k-i+1}$. Therefore, the load on each machine in $\{G_1, \ldots, G_k\}$ in exactly 2 (where the load on the machine in G_0 is 0 since it has no jobs to process). From here, we get that the PoA is at least order of $\frac{\log m}{\log \log m}$. $\square$

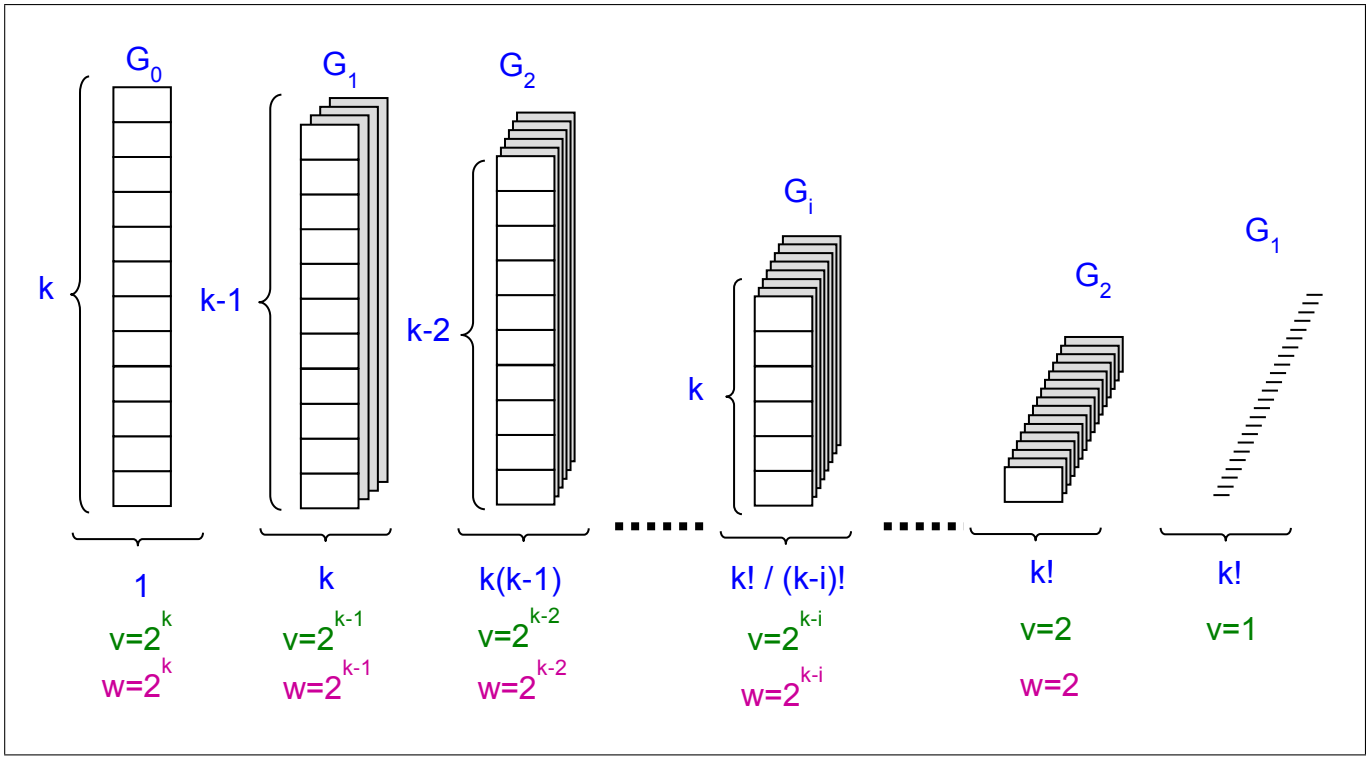


Figure 2.9: An example of an inefficient equilibrium for the case of unrelated machines.

2.3 References

[CorreaJ05] Correa, J., A. Schulz, and N. Stier-Moses, On the inefficiency of equilibria in congestion games. Integer Programming and Combinatorial Optimization, 2005: p. 171-177.

[CorreaJR10] Correa, J.R. and N.E. Stier-Moses, Wardrop equilibria. Wiley Encyclopedia of Operations Research and Management Science, 2010.

[EaselyD10] Easley, D. and J. Kleinberg, Networks, Crowds, and Markets: Overview; Part I. Graph Theory and Social Networks: 2. Graphs; 3. Strong and weak ties; 4. Networks in their surrounding contexts; 5. Positive and negative relationships; Part II. Game Theory: 6. Games; 7. Evolutionary game theory; 8. Modeling network traffic using game theory; 9. Auctions; Part III. Markets and Strategic Interaction in Networks: 10. Matching markets; 11. Network models of markets with intermediaries; 12. Bargaining and power in networks; Part IV. Information Networks and the World Wide Web: 13. The structure of the Web; 14. Link analysis and Web search; 15. Sponsored search markets; Part V. Network Dynamics: Population Models: 16. Information cascades; 17. Network effects; 18. Power laws and rich-get-richer phenomena; Part VI. Network Dynamics: Structural Models: 19. Cascading behavior in networks; 20. The small-world phenomenon; 21. Epidemics; Part VII. Institutions and Aggregate Behavior: 22. Markets and information; 23. Voting; 24. Property. 2010: Cambridge University Press.