CS 573: Algorithmic Game Theory	Lecture date: January 25th, 2008
Instructor: Chandra Chekuri	Scribe: Bolin Ding

Contents

1 Introduction to Inefficiency of Equilibrium

In this section, we will introduce two concepts *price of anarchy* (PoA) and *price of stability* (PoS) to quantify how much selfish players hurt the system. We assume selfish people employ pure strategies to maximize his/her payoff, and thus only a pure Nash equilibrium is a stable status of the system for selfish people. We will also use *load balancing games* as examples to illustrate how to analyze PoA and PoS.

1.1 Price of Anarchy and Price of Stability

Given a game $(N, < S_i >, < u_i >)$, where $N = \{1, ..., n\}$ is the set of n players, S_i is the set of possible strategies of player i, and $u_i : S = S_1 \times ... \times S_n \mapsto R$ is the payoff function of player i. Define $A = \{s \in S | s \text{ is a pure Nash equilibrium}\}$. To define PoA and PoS, we need to measure the "value" of strategies picked by players. Let $f : S \mapsto R^+$ be the *social objective function*, and s^* be a *social optimum* iff $f(s^*) = \max_{s\in S} f(s)$ (the objective is to maximize $f(s)$) or $f(s^*) = \min_{s\in S} f(s)$ (the objective is to minimize $f(s)$).

Definition 1.1 (Price of Anarchy (PoA)) *The price of anarchy is defined as*

$$\begin{aligned} PoA &= \max_{s\in A} \frac{f(s^*)}{f(s)}, \quad \text{if the objective is to maximize } f(s) \\ & \max_{s\in A} \frac{f(s)}{f(s^*)}, \quad \text{if the objective is to minimize } f(s). \end{aligned}$$

Definition 1.2 (Price of Stability (PoS)) *The price of stability is defined as*

$$\begin{aligned} PoS &= \min_{s\in A} \frac{f(s^*)}{f(s)}, \quad \text{if the objective is to maximize } f(s) \\ & \min_{s\in A} \frac{f(s)}{f(s^*)}, \quad \text{if the objective is to minimize } f(s). \end{aligned}$$

Intuitively, PoA is the worst case of how much selfish people may hurt the social optimum, because selfish people can make the system stable in any pure Nash equilibrium $s \in A$. On the other hand, PoS is the best case one can expect selfish people to achieve a Nash equilibrium, which hurts the social optimum as least as possible (a ratio of PoS). Note PoA and PoS can be extended to mixed Nash equilibrium, but we will first focus on pure Nash equilibrium in this section.

1.2 Example: Load Balancing Games

We will use *load balancing games* as examples to show how to analyze PoA and PoS of a game. First, Let's define a load balancing game as follows.

There are m machines available, and n jobs controlled by n *selfish players* $1, ..., n$. Player i controls job i with weight $w_i \geq 0$, which needs to be scheduled on one of the m machines. Define a load balancing game as $(N, < S_i >, < u_i >)$, where $N = \{1, .., n\}$ denotes n players (jobs), and $S_i = \{1, ..., m\}$. In a vector of strategies $s \in S = S_1 \times ... \times S_n$, entry $s_i \in S_i$ means job i is scheduled on machines s_i. Define the *load* on machine j as the total weight of jobs scheduled on machine j

$$l_j(s) = \sum_{i:s_i=j} w_i. \tag{1}$$

In a general setting, define $u_i(s) = f_i(l_{s_i}(s))$ as the cost of player i w.r.t. the vector of strategies s, where $f_i(x)$ is an ascending monotone function. The objective of each player is to *minimize* $u_i(s)$.

Second, we want to *minimize* the *social objective function*,

$$f(s) = \max_{1\leq j\leq m} l_j(s), \tag{2}$$

which denotes the time when the last job is completed (*makespan*) in scheduling problems.

Note Equation (1) assumes all machines processes jobs at the same speed. We will discuss a more general setting of load balancing games, where different machines are with different job-processing speeds in Section 1.2.3.

1.2.1 Existence of Nash Equilibrium of Pure Strategies

We prove there exists a pure Nash equilibrium in every load balancing game.

Lemma 1.3 *If each function $u_i(s) = f_i(l_{s_i}(s))$ is ascending monotone w.r.t. $l_{s_i}(s)$, then a load balancing game has a Nash equilibrium of pure strategies.*

Proof: To prove the existence of a pure Nash equilibrium, we define a *potential function* $\Phi : S \mapsto X$, s.t. i) X is a *totally ordered set*; ii) $\Phi(s') < \Phi(s)$ if there exists i and $s'_i \neq s_i$ s.t. $u_i(s'_i, s_{-i}) < u_i(s_i, s_{-i})$ (i.e. s is not a Nash equilibrium). Because set $S = S_1 \times ... \times S_n$ is finite in load balancing games, the construction of function Φ satisfying properties i) and ii) immediately implies the existence of a pure Nash equilibrium (there exists $s_0 \in S$ s.t. $\Phi(s_0) = \inf_{s\in S} \Phi(s)$).

In load balancing games, for $s \in S$, let $\Phi(s) = (\alpha_1, ..., \alpha_m)$, where $\alpha_1 \geq ... \geq \alpha_m$ is the sorted list of loads $\{l_1(s), ..., l_m(s)\}$ on machines. It can be verified that

$$X = \{(\alpha_1, ..., \alpha_m)|\alpha_1 \geq ... \geq \alpha_m \geq 0, \sum_{j=1}^{m} \alpha_j = \sum_{i=1}^{n} w_i\}$$

is totally ordered, with '<' defined as the *lexicographic order*, for example, $(2,2,2) < (3,2,1)$. Therefore, to prove the existence of a pure Nash equilibrium, we only need to show that $\Phi(s') < \Phi(s)$ if there exists i and $s'_i \neq s_i$ s.t. $u_i(s'_i, s_{-i}) < u_i(s_i, s_{-i})$. We leave it as an exercise to readers. □

Note the proof of Lemma 1.3 can be easily extended to a more general setting, where different machines are with different job-processing speeds as introduced in Section 1.2.3

1.2.2 Analysis of PoA and PoS

Theorem 1.4 *In a load balancing game, given the load on a machine and the social objective function defined in Equation* (1) *and* (2), *respectively, we have* $PoA \le 2 - 2/(m+1)$. *Moreover, there exists an instance of load balancing game, s.t.* $PoA = 2 - 2/(m+1)$.

Proof: Let the social optimum be

$$OPT = f(s^*) = \min_{s \in S} f(s) = \min_{s \in S} \max_{1 \le j \le m} l_j(s).$$

We can prove the following three claims:

$$OPT \ge \max_{1 \le i \le n} w_i \quad (3)$$

$$OPT \ge \frac{1}{m} \sum_{1 \le i \le n} w_i \quad (4)$$

$$\text{Vector of strategies } s \in S \text{ is a pure Nash equilibrium} \Leftrightarrow$$
$$\forall j, j', i \text{ s.t. } s_i = j: \quad l_j(s) \le l_{j'}(s) + w_i. \quad (5)$$

(3) and (4) are straightforward. (5) is because of the definition of Nash equilibrium — if job i can be scheduled on machine j', instead of j, to get a lower load $l_{j'}(s) + w_i (< l_j(s))$ on machines j', then s is not a Nash equilibrium.

Consider any Nash equilibrium $s \in A$. W.l.o.g., we can suppose (i) $s_1 = 1$ (i.e., job 1 is scheduled on machine 1), (ii) $l_1(s) = \max_j l_j(s) = f(s)$, and (iii) $w_1 = \min\{w_i | s_i = 1\}$. By (5), we have

$$\begin{aligned} l_1(s) &\le l_2(s) + w_1 \\ l_1(s) &\le l_3(s) + w_1 \\ &\dots \quad \dots \\ l_1(s) &\le l_4(s) + w_1. \end{aligned}$$

Sum them up, then we have $(m-1)l_1(s) \le l_2(s) + ... + l_m(s) + (m-1)w_i$, and thus $ml_1(s) \le l_1(s) + l_2(s) + ... + l_m(s) + (m-1)w_i$. By $\sum_j l_j(s) = \sum_i w_i$, we have

$$\begin{aligned} ml_1(s) &\le \sum_{1 \le i \le n} w_i + (m-1)w_1 \\ \Rightarrow l_1(s) &\le \frac{1}{m} \sum_{1 \le i \le n} w_i + \frac{m-1}{m} w_1 \\ \Rightarrow l_1(s) &\le OPT + \frac{m-1}{m} w_1 \quad \text{from (4).} \quad (6) \end{aligned}$$

We consider two cases:

- If $|\{w_i|s_i = 1\}| = 1$ (only job 1 is scheduled on machine 1), we can prove $l_1(s) = OPT$.
- If $|\{w_i|s_i = 1\}| > 1$, we have

$$\begin{aligned} w_1 &\leq \frac{1}{2}l_1(s) \quad \text{from } w_1 = \min\{w_i|s_i = 1\} \\ \Rightarrow \quad l_1(s) &\leq OPT + \frac{m-1}{2m}l_1(s) \quad \text{from (6)} \\ \Rightarrow \quad l_1(s) &\leq \frac{2m}{m+1}OPT \\ &\leq (2 - \frac{2}{m+1})OPT. \end{aligned} \tag{7}$$

Note if $|\{w_i|s_i = 1\}| = 1$, (7) is still true. Therefore, from (7) and $l_1(s) = \max_j l_j(s) = f(s)$, we already prove: *for any Nash equilibrium* $s \in A$, $f(s) \leq (2 - \frac{2}{m+1})OPT$, and thus, PoA $\leq 2 - \frac{2}{m+1}$.

We need also show our analysis is tight, i.e., there exists an instance of load balancing game and a Nash equilibrium $s \in A$ for this game, s.t. $\frac{f(s)}{OPT} = 2 - \frac{2}{m+1}$, to complete our proof. Following is such an example: There are $n = m(m-1) + 2 = m^2 - m + 2$ jobs and m machines. $w_1 = w_2 = m$ and $w_i = 1$ for $i = 3, ..., m^2 - m + 2$. In this instance, $OPT = m + 1$. There is a Nash equilibrium s, where job 1 and job 2 are scheduled on machine 2, and m jobs are scheduled on each of other $m - 1$ machines. We have $f(s) = 2m$, and thus $\frac{f(s)}{OPT} = 2 - \frac{2}{m+1}$. □

Theorem 1.5 *In a load balancing game, given the load on a machine and the social objective function defined in Equation* (1) *and* (2)*, respectively, we have* $PoS = 1$.

Proof: Recall in the proof of Lemma 1.3, we construct a *potential function* $\Phi : S \mapsto X$, s.t. i) X is a *totally ordered set*; ii) $\Phi(s') < \Phi(s)$ if there exists i and $s'_i \neq s_i$ s.t. $u_i(s'_i, s_{-i}) < u_i(s_i, s_{-i})$ (i.e. s is not a Nash equilibrium). From the definition of $f(s)$ and the construction of $\Phi(s)$, we can further prove iii) $f(s') \leq f(s)$ if $\Phi(s') < \Phi(s)$.

Let the social optimum be

$$OPT = f(s^*) = \min_{s \in S} f(s).$$

If $s^0 = s^*$ is a Nash equilibrium, we have completed our proof because $\frac{f(s^0)}{f(s^*)} = 1$. Otherwise there exists s^1 s.t. $\Phi(s^1) < \Phi(s^0)$ and thus $f(s^1) \leq f(s^0)$. Similarly, if s^1 is a Nash equilibrium, we have completed our proof. Otherwise, we can find s^2 s.t. $\Phi(s^2) < \Phi(s^1)$... This process will finally terminate because $S = S_1 \times ... \times S_n$ is finite, and suppose s^x is the Nash equilibrium we finally find s.t. $f(s^x) \leq f(s^{x-1}) \leq ... \leq f(s^0) = f(s^*)$. Obviously, PoS ≥ 1. Therefore, we must have $f(s^x) = f(s^*)$ and thus PoS $= 1$. □

1.2.3 A More General Setting

In a more general setting, different machines are with different job-processing speeds, we modify (1) as follows:

$$l_j(s) = \frac{1}{v_j} \sum_{i:s_i=j} w_i, \tag{8}$$

where v_j denotes the job-processing speed of machine j.

Theorem 1.6 *In a load balancing game, given the load on a machine and the social objective function defined in Equation* (8) *and* (2), *respectively, we have* $PoA \in O(\log m/\log\log m)$. *Moreover, there exists an instance of load balancing game, s.t.* $PoA \in \Omega(\log m/\log\log m)$.

Let $v_{\max} = \max\{v_1, ..., v_n\}$ and $v_{\min} = \min\{v_1, ..., v_n\}$ be the maximum and the minimum job-processing speeds, respectively. We can prove PoA $\in O(\log m/\log\log m)$, and there exists an instance of load balancing game, s.t. PoA $\in \Omega(\log m/\log\log m)$. Details are omitted here.

Theorem 1.7 *In a load balancing game, given the load on a machine and the social objective function defined in Equation* (8) *and* (2), *respectively, we have* $PoS = 1$.

The proof of Theorem 1.5 can be simply extended to prove Theorem 1.7.