601.436/636 Algorithmic Game Theory **Lecturer:** Michael Dinitz
Topic: Load Balancing Game **Date:** 3/26/20
Scribe: Michael Dinitz

15.1 Introduction

The last few weeks we've primarily focused on pure Nash equilibria. This is because we've been talking a lot about potential games (where they always exist) and about smooth games) where bounds on the price of anarchy against pure Nash generalize all the way to bounds on the pricet of total anarchy (against coarse correlated equilibria). But there are some interesting and natural games where pure Nash exhibit fundamentally different behavior even from mixed Nash, even though both exist. These games are clearly not smooth, but are still very important. Today we'll talk about one of the most natural of these games, which is focused on *load balancing*.

15.2 Game Definition

- There are n jobs, where job i has weight w_i. These are the players (or, equivalently, each player controls one of these jobs).
- There are m machines, and these machines are the pure strategies for each player (i.e., $S_i = [m]$ for all $i \in [n]$). So S (the set of strategy profiles) is $[m]^n$, and we can think of any particular strategy profile as an assignment $A : [n] \to [m]$.
- The *load* of machine j under assignment A is $\ell_j(A) = \sum_{i \in [n]: A(i)=j} w_i$.
- The cost to player i of assignment A is $c_i(A) = \ell_{A(i)}(A)$: the load of the machine on which player i is assigned.
- The global notion of cost we will consider is the *makespan*, i.e., the maximum load: $\text{cost}(A) = \max_{j \in [m]} \ell_j(A)$.

From a pure optimization view, this problem is known as "makespan scheduling on identical machines": find the assignment of jobs to machines that minimizes the time at which all jobs are finished, which is clearly the time at which the most loaded machine finishes, known as the makespan. Scheduling is a huge area of algorithms, and there are a ton of variants. For example, we might care about things other than the makespan, like the sum of completion times of the jobs (in which case the ordering of the jobs on each machine matters, unlike for makespan). Or the machines might not be identical: they might be *related*, where each machine j has a speed s_j and the time to complete job i on machine j is w_i/s_j. Or they might be *unrelated*, where each job i has some arbitrary processing time for each machine, so we have weights w_{ij} (the time to complete job i on machine j) instead of just w_i. All of these optimization problems have game theoretic analogues, but we won't go into them. Chapter 20 of NRTV goes into the related machine setting a little bit, but there's obviously a huge amount more out there.

15.3 Basic Results and Definitions

We're going to want to argue that the pure Nash of this game behave very differently from the mixed Nash. So first, we should establish that there are indeed pure Nash equilibria. To do this, we'll use an argument which is *almost* a potential function argument, but isn't quite. We won't assign values to strategy profiles, but will instead find an ordering of strategy profiles so that players who want to deviate always cause us to go down in this ordering. This will then imply that the minimum of this ordering is a pure Nash.

To define this ordering, note that each assignment A results in a *sorted load vector* $\lambda(A) = (\lambda_1(A), \lambda_2(A), \ldots, \lambda_m(A))$, where $\lambda_i(A)$ is the load of the i'th most loaded machine (breaking ties arbitrarily). In other words, we get $\lambda(A)$ by sorting the vector $(\ell_1(A), \ell_2(A), \ldots, \ell_m(A))$.

Now we can order the sorted load vectors *lexicographically.* Recall that in a lexicographic ordering, we say that a vector $x = (x_1, \ldots, x_m)$ is less than a vector $y = (y_1, \ldots, y_m)$ if, in the first index in which they differ, x is less than y. Slightly more formally, $x \prec y$ if there is some $i \leq m$ such that $x_j = y_j$ for all $j < i$ and $x_i < y_i$. Note that this is a total ordering: for any two different vectors (of the same length), one of them is less than the other.

This induces an ordering on assignments: we say that $A \prec B$ if $\lambda(A) \prec \lambda(B)$. With these definitions in hand, we can now prove that there is some pure Nash.

Theorem 15.3.1. *The load balancing game always has a pure Nash equilibrium.*

Proof. Note that there are only a finite number of assignments (exactly m^n), so there is some assignment A which is minimum according to the ordering above ($A \prec A'$ for all assignments $A' \neq A$, i.e., A has the lexicographically smallest sorted load vector). We claim that A is a pure Nash equilibrium.

To see this, let's suppose that some player i has incentive to deviate and then derive a contradiction. Let j be the index of $A(i)$ in the sorted vector (so i is currently on the j'th most loaded machine), and let k be the index in the sorted load vector of the machine that it wants to deviate to. Then after deviating, the machine that i deviates to must have load less than $\lambda_j(A)$. So all machines other than j and k are unchanged, and the load on j and k are both less than the load on j before deviation. Thus if A' is the assignment after deviating, $\lambda_i(A') = \lambda_i(A)$ for all $i < j$ and $\lambda_j(A') < \lambda_j(A)$. Hence $A' \prec A$, contradicting our assumption that A was the minimum assignment according to the ordering.

So no player can have incentive to deviate, and so A is a pure Nash equilibrium. $\square$

This immediately gives the following corollary:

Corollary 15.3.2. *The Price of Stability is* 1.

Proof. Let A be the pure Nash we found above: the assignment minimizing the lexicographic ordering. Then A is a pure Nash, and we claim that A is also optimal with respect to the global cost function (the makespan). This is trivial to see: let A^* be any other assignment. Then

$\lambda_1(A) \le \lambda_1(A^*)$ by the definition of A. But λ_1 is exactly the load on the most loaded machine, i.e., our cost function! So $\text{cost}(A) \le \text{cost}(A^*)$. □

15.4 Pure vs Mixed Nash: Simple Example

Unlike most of the games that we've been talking about recently, in load balancing games mixed Nash can be much worse than pure Nash. Let's first see an example where they're a little bit different, and then we'll analyze each type of equilibrium in turn.

Let $m = n$ and let $w_i = 1$ for all $i \in [n]$. Then any pure Nash equilibrium assigns each job to a different machine, so the pure price of anarchy is 1. But now consider the product distribution σ over S obtained by having each player choose a machine uniformly at random. This is actually a mixed Nash equilibrium, since

$$\mathop{\mathbf{E}}_{A\sim\sigma}[c_i(A)] = 1 + \sum_{k\neq i} \mathbf{Pr}[A(k) = A(i)] = 1 + \frac{n-1}{n}$$

and

$$\mathop{\mathbf{E}}_{A\sim\sigma}[c_i(A_{-i}, j)] = 1 + \sum_{k\neq i} \mathbf{Pr}[A(k) = j] = 1 + \frac{n-1}{n}$$

for all $j \in [n]$. Thus player i has no incentive to deviate, so σ is a mixed Nash. But it's not hard to see that $\mathbf{E}[\text{cost}(A)] > 1$. For example, if $n = m = 2$ then there are four possible assignments, two of which have cost 1 and two of which have cost 2, and thus the expected cost is $3/2$.

So this example shows that there can be a gap. But maybe that's just this example, and in general the bound on the pure price of anarchy is the same as the bound on the price of anarchy? This turns out not to be true, and is the focus of most of the rest of today: we'll prove that the pure price of anarchy is always at most $2 - \frac{2}{m+1}$ (which isn't as good as the 1 from the example, but it still quite small) while the price of anarchy can be $\Omega\left(\frac{\log m}{\log\log m}\right)$.

15.5 Pure Price of Anarchy

Let's prove the main upper bound.

Theorem 15.5.1. *The pure price of anarchy is at most* $2 - \frac{2}{m+1}$.

Proof. Let A be a pure Nash equilibrium, and let j^* be the machine with highest load under A (so $\text{cost}(A) = \ell_{j^*}(A)$). Let i^* be the job with smallest weight that A assigns to j^*.

If i^* is the *only* job assigned to j^*, then since the optimal solution has to assign i^* somewhere we know that $OPT \ge w_{i^*} = \text{cost}(A)$, and thus A is actually optimal and we are done. So without loss of generality, we may assume that there are at least two jobs assigned to j^*, and thus $w_{i^*} \le \frac{1}{2}\ell_{j^*}(A) = \frac{1}{2}\text{cost}(A)$.

Since A is a pure Nash, player i^* does not want to switch from j^* to any other machine, and thus

$$\ell_j(A) \ge \ell_{j^*}(A) - w_{i^*} \ge \text{cost}(A) - \frac{1}{2}\text{cost}(A) = \frac{1}{2}\text{cost}(A) \qquad \text{for all } j \neq j^*.$$

Now let's use this inequality together with the fact that the most loaded machine has load at least the *average* load to get that

$$\begin{aligned} OPT &\geq \frac{\sum_{i=1}^n w_i}{m} = \frac{\sum_{j=1}^m \ell_j(A)}{m} = \frac{\ell_{j^*}(A) + \sum_{j \neq j^*} \ell_j(A)}{m} \\ &\geq \frac{\text{cost}(A) + \sum_{j \neq j^*} \frac{1}{2}\text{cost}(A)}{m} = \frac{\left(1 + \frac{m-1}{2}\right)\text{cost}(A)}{m} = \frac{\left(\frac{m+1}{2}\right)\text{cost}(A)}{m} \\ &= \left(\frac{m+1}{2m}\right)\text{cost}(A). \end{aligned}$$

Now we can rearrange this to get

$$\frac{\text{cost}(A)}{OPT} \leq \frac{2m}{m+1} = 2 - \frac{2}{m+1}$$

as claimed. $\square$

This analysis is exactly tight: there is an example of a load balancing game in which there is a pure Nash equilibrium that is $2 - \frac{2}{m+1}$ times worse than the optimum. This game is pretty simple. We have $n = 2m$ with m jobs of weight m and m jobs of weight 1. Then OPT is $m+1$: for every machine we can assign one job of weight m and one job of weight 1. But consider the assignment in which two large jobs choose machine 1, the remaining $m-1$ large jobs choose machines $2, 3, \ldots, m-1$, and all the small jobs choose machine m. Then no one has any incentive to switch, but the cost is equal to $2m$. Thus the pure price of anarchy in this game is at least

$$\frac{2m}{m+1} = 2 - \frac{2}{m+1}.$$

15.6 Mixed Nash

Now let's think about mixed Nash. We first want to show that they can be significantly worse than pure Nash. To do this, let's just expand on our simple example: let's set $n = m$ and $w_i = 1$ for all $i \in [n]$.

Theorem 15.6.1. *The price of anarchy is at least* $\Omega\left(\frac{\ln m}{\ln \ln m}\right)$.

Proof. We know from our previous discussion that OPT is 1 in this very simple game (each job gets its own machine), and that each player choosing a machine uniformly at random is a mixed Nash σ. So we just want to show that the expected cost of σ is at least $\Omega((\frac{\ln m}{\ln \ln m})$.

This turns out to be a classic "balls-in-bins" problem from probability theory. In the "balls-in-bins" setting there are n balls and m bins, and each ball is thrown into a bin independently and uniformly at random. Probabilists have extensively analyzed this process and there is a ton known about this distribution. For our purposes, what we care about is the "expected maximum occupancy", which is just the expectation of the load of the most loaded bin. In other words, exactly $\mathbf{E}_{A \sim \sigma}[\text{cost}(A)]$.

A classical result is that the expected maximum occupancy is $\Theta\left(\frac{\ln m}{\ln(1 + \frac{m}{n}\ln m)}\right)$. If we plug in our situation of $m = n$, this gives an expected maximum occupancy of $\Theta\left(\frac{\ln m}{\ln \ln m}\right)$ as claimed. $\square$

So now we know that the price of anarchy is significantly worse than the pure price of anarchy in load balancing games. This is the basic point of this lecture. But to complete our analysis of these games, we might also want an upper bound on the price of anarchy. It turns out that the above example is actually the worst case. We don't have the time, so I'll just give a sketch of the proof of the following theorem.

Theorem 15.6.2. *In any load balancing game, the price of anarchy is at most* $O\left(\frac{\ln m}{\ln\ln m}\right)$.

Proof Sketch. Let σ be a mixed Nash equilibrium. We want to bound $\mathbf{E}_{A\sim\sigma}[\max_{j\in[m]} \ell_j(A)]$, but the first step is to bound $\max_{j\in[m]} \mathbf{E}_{A\sim\sigma}[\ell_j(A)]$. This turns out to be pretty easy: we can basically just repeat the analysis of pure Nash. In other words, if we fix any $j \in [m]$, the same basic argument as before implies that

$$\mathop{\mathbf{E}}_{A\sim\sigma}[\ell_j(A)] \le \left(2 - \frac{2}{m+1}\right)\cdot OPT.$$

Of course the analysis changes slightly (we have to let j^* be the machine with the maximum *expected* load and let i^* be the job of minimum weight that has a nonzero probability of choosing j^*), but it's basically the same thing so I'll skip it.

So now we know that every machine has expected load that's at most twice OPT. Now we can prove *concentration.* fix some machine j. Since the load on machine j is a random variable which is just the sum of independent Bernoulli random variables (representing for each ball whether it chooses machine j or not), we can apply a *Chernoff bound* to the load. We'll use a simple Chernoff bound for this sketch, which if you're interested you can find in Section 1.6 of [DP09]:

Theorem 15.6.3. *If* $X = X_1 + X_2 + \cdots + X_n$ *where each* X_i *is an independent random variable in* $[0,1]$ *(not necessarily from the same distribution), then for all* $t > 2e\cdot\mathbf{E}[X]$ *we have* $\mathbf{Pr}[X > t] \le 2^{-t}$.

Now this theorem implies that $\mathbf{Pr}[\ell_j(A) > t] \le 2^{-t}$ for all $t > 4e\cdot OPT$. For any $c \ge 0$, if we set $t = (c\ln m)OPT$ (which is larger than $4e\cdot OPT$ for large enough m) then this implies that $\mathbf{Pr}[\ell_j(A) > (c\ln m)OPT] \le \frac{1}{m^c}$. Now a union bound over all m machines implies that

$$\mathbf{Pr}[\mathrm{cost}(A) > (c\ln m)OPT] = \mathbf{Pr}\left[\max_{j\in[m]} \ell_j(A) > (c\ln m)OPT\right] \le \frac{1}{m^{c-1}}.$$

Now we can set $c = 1$ and integrate over the tail to get

$$\mathbf{E}[\mathrm{cost}(A)] \le \ln m\cdot OPT + \int_1^\infty (x\ln m)OPT\frac{1}{m^{x-1}}dx \le O(\ln m)\cdot OPT$$

This is a little weaker than the actual theorem: it's a bound of $O(\ln m)$ instead of $O\left(\frac{\ln m}{\ln\ln m}\right)$. To get the full theorem, we just need to use a slightly stronger (but more complicated and less intuitive) version of the Chernoff bound. Details are in NRTV Chapter 20. ☐

References

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, USA, 1st edition, 2009.