Stackelberg Scheduling Strategies

Tim Roughgarden
Cornell University
Machines and Latencies

The Model:

- \( m \) machines 1, 2, ..., \( m \)
- A rate \( r \) of jobs
  - jobs are small (model in a cts way)
- For each machine \( i \), a load-dependent latency function \( l_i(\cdot) \)
  - assume continuous, nondecreasing

Example: (\( r = 1 \))

- jobs have latency 1
  \[ l_1(x) = 1 \]
- jobs have latency \( \frac{1}{4} \)
  \[ l_2(x) = x^2 \]
Equilibria

Job Assignments:

• $x_i$ = amount of jobs assigned to machine $i$

• vector $x \iff$ assignment of all jobs

Equilibria:

• which job assignments are “stable”?  
  - jobs controlled by selfish, noncooperative agents 
  - no job wants to switch machines (no job should be envious)
More on Equilibrium Assignments

\[ l_1(x) = 1 \quad l_2(x) = x^2 \]

**Def:** an assignment is at Nash equilibrium (is a Nash assignment) if:
- all used machines have equal latency
- unused machines have greater latency

**Fact:** always have existence, uniqueness
How Good is an Assignment?

The Cost of an Assignment:

- $C(x) = \text{cost or total latency experienced by assignment } x$:
  \[ \sum_i x_i \cdot l_i(x_i) \]

⇒ our notion of system performance

Example:

\[
\begin{align*}
\frac{1}{2} & \quad \frac{1}{2} \\
1 & \quad x^2
\end{align*}
\]

\[
\text{cost} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{4} = ?
\]
How Good are Nash Assignments?

**Goal:** prove that Nash assignments are near-optimal
- want a laissez faire approach to regulating users

**Problem:** false in general!

Example: \((r=1, k \text{ large})\)

Nash costs 1  
\[1 \times^k\]  
OPT costs \(\approx 0\)
Near-Optimal Nash Assignments

Solution #1: relax our notion of “near-optimal”

- Theorem [RT00]: cost of Nash assignment = cost of OPT at rate $2r$

Solution #2: restrict to linear latency functions ($l_i(x) = a_i x + b_i$)

- Theorem [RT00]: cost of Nash assignment = $\frac{4}{3}$ times cost of OPT (at same rate $r$)
Taming Selfishness through a Manager

Solution #3: change the model!

• not all jobs need be controlled by selfish users
  - “centrally controlled” vs. “selfishly controlled” jobs
  - behavior of selfish users depends on assignment of managed jobs

Goal:

• assign centrally controlled jobs to induce “good” selfish behavior
  - see also [Korlis, Lazar, Orda 97]
Examples

Strategy #1:

\[ \begin{array}{c}
\text{Strategy} \\
0 \\
1
\end{array} \quad \begin{array}{c}
\frac{1}{2} \\
\times
\end{array} \quad \begin{array}{c}
\text{Induced Equilibrium} \\
(\text{same as Nash eq})
\end{array} \]

\[ \begin{array}{c}
\text{Strategy} \\
0 \\
1
\end{array} \quad \begin{array}{c}
\frac{1}{2} \\
\times
\end{array} \quad \begin{array}{c}
\frac{1}{2} \\
\frac{1}{2}
\end{array} \]

Strategy #2:

\[ \begin{array}{c}
\text{Strategy} \\
\frac{1}{2} \\
\times
\end{array} \quad \begin{array}{c}
0 \\
1
\end{array} \quad \begin{array}{c}
\text{Induced Equilibrium} \\
(\text{same as optimum!})
\end{array} \]

\[ \begin{array}{c}
\frac{1}{2} \\
\times
\end{array} \quad \begin{array}{c}
\frac{1}{2} \\
\frac{1}{2}
\end{array} \]
Examples

Any Strategy:

$\begin{array}{c}
\text{1} \\
\text{2x}
\end{array}$

Strategy

$\begin{array}{c}
\frac{1}{2} \\
\frac{1}{2}
\end{array}$

Induced Equilibrium (same as Nash eq)

Optimal Assignment (cost = ?):

$\begin{array}{c}
\frac{3}{4}
\end{array}$

1

$\begin{array}{c}
\frac{1}{4}
\end{array}$

2x
Stackelberg Strategies

• Stackelberg strategy = assignment of centrally controlled jobs
  ⇒ yields an induced equilibrium

• Basic Questions:
  • what’s the best strategy?
    • can we compute/characterize it?
  • how inefficient is the best induced equilibrium?
    • are we provably near-optimal?
Our Results - General Latency Functions

**Theorem 1:** Can efficiently compute a strategy inducing an equilibrium with cost $= \frac{1}{\beta} \times \text{cost of optimal assignment}$.

$\beta = \text{fraction of centrally assigned jobs}$

**Note:**

$\left(\frac{1}{\beta}\right) \times \text{OPT}$ is best possible

$\begin{array}{c}
\frac{1}{\beta} \\
\beta \\
\end{array}$

$\begin{array}{c}
\frac{x}{(1-\beta)}^k \\
1-\beta \\
\end{array}$

(Induced eq = Nash eq for any strategy)
Our Results - Linear Latency Functions

Theorem 2: Can efficiently compute a strategy inducing an equilibrium with cost = \( \frac{4}{3+\beta} \) × cost of optimal assignment.

\( \beta = \text{fraction of centrally assigned jobs} \)

Note:
\( \frac{4}{3+\beta} \) × OPT is best possible

\[
\begin{array}{c|c|c}
1 & \beta & \frac{x}{1-\beta} \\
\end{array}
\]

(Induced eq = Nash eq for any strategy)
What makes a strategy (in)effective?

The Aloof Strategy

• ignore selfish users, assign centrally controlled jobs in cheapest possible way

\[
\begin{array}{c|c}
0 & 1 \\
\hline \\
\frac{1}{2} & x \\
\end{array}
\quad\rightarrow\quad
\begin{array}{c|c}
0 & 1 \\
\hline \\
1 & x \\
\end{array}
\]

Strategy

Induced Equilibrium (same as Nash eq)

• ⇒ induced eq costs 1, OPT = $\frac{3}{4}$
What makes a strategy (in)effective? (con’d)

The Scale Strategy

• compute optimal assignment $x$ of all jobs, assign centrally controlled jobs via $\beta \cdot x$

\[ \begin{align*}
\text{Strategy} & \quad 1 \quad (3/2)x \\
\text{Induced Equilibrium} & \quad 1 \quad (3/2)x
\end{align*} \]

• $\Rightarrow$ induced eq costs 1, $\text{OPT} = 5/6$
What have we learned?

**Morals:**

- **avoid** machines that selfish users will (over)use anyways
- assign centrally controlled jobs to machines **least attractive** to the selfish users
  \(\Leftrightarrow\) edges with large latency
- **don’t oversaturate** any machines
The LLF Strategy

Largest Latency First (LLF):

• compute opt assignment $x$ of all jobs

• for each machine $i$, in order of decreasing $l_i(x_i)$ values:
  - if $\geq x_i$ centrally controlled jobs remain, assign $x_i$ jobs to $i$ (saturate machine $i$)
  - else assign remaining jobs (if any) to $i$
Example

Optimal Assignment: \((r=1)\)

\[ l_1 = 1 \]
\[ l_2 = \frac{1}{2} \]
\[ l_3 = \frac{1}{2} \]

The LLF Strategy: \((\beta = \frac{1}{2})\)

\[ 5/12 \]
\[ 1 \]
\[ 2x \]
\[ (3/2)x \]

\[ 1/4 \]
\[ 1/12 \]
\[ 0 \]
LLF with General Latency Functions

**Theorem 1:** The LLF strategy induces an equilibrium with cost
\[ \frac{1}{\beta} \times \text{cost of optimal assignment.} \]

**Proof idea:** Use iterative structure of LLF to proceed by induction on # of machines.
Proof Idea

Induced Equilibrium (w.r.t. LLF):

$64K \text{ Question: Is our inductive guarantee strong enough?}$
Proof of Theorem 1

machines have latency $\geq L$

- $\mu = \# \text{ of jobs}$
- $\text{OPT pays}$
  - $\geq \text{we pay}$
  - $\geq \mu L$

all machines same latency $= L$

- we pay $(1-\mu)L$
- $\beta' = \frac{(\beta-\mu)}{(1-\mu)}$
- $\text{IH} \implies \text{OPT pays}$
  - $\geq \beta' \times (1-\mu)L$
  - $= (\beta-\mu)L$
Bug Fix

Problem: Can’t recurse!

Fix: LLF failed to saturate machine 1

- Machine 1 is high-latency \((\geq L)\)
- Our solution costs \(L\) (assume \(r=1\))
- OPT puts \(\geq \beta\) jobs on machine 1
  \[\Rightarrow\] OPT pays \(\geq \beta L\)
Bug Fix (con’d)

**Problem:** have we really covered all the cases?

An LLF Strategy

\[
\begin{align*}
\text{1/3} & \quad \text{1/6} \\
\text{1} & \quad \text{1} \\
\text{An LLF Strategy}
\end{align*}
\]

An Induced Equilibrium

\[
\begin{align*}
\text{2/3} & \quad \text{1/3} \\
\text{1} & \quad \text{1} \\
\text{An Induced Equilibrium}
\end{align*}
\]

**Lemma:** Only a problem with locally constant latency functions.

- can reassign selfish users, get a "nicer" induced equilibrium
LLF with **Linear** Latency Functions

**Theorem 2:** The LLF strategy induces an equilibrium with cost
\[ \frac{4}{3+\beta} \times \text{cost of optimal assignment}. \]

**Main difficulty:**
- previous argument too weak for:
- need detailed study of Nash, OPT when all latencies are linear
Computing Optimal Strategies

**Question:** is the LLF strategy optimal on all instances?

**Bad news:** no, in fact:

**Theorem 3:** Computing the optimal strategy is **NP-hard** (even for linear latency fns).

- **Compare to:** optimal, Nash assignments
Sketch of Reduction

**Idea:** view the best SS as solving the optimization problem $P$:

$$\max \sum_i \max (0, x_i - \text{Nash}_i)$$
$$\text{s.t.: } \sum_i x_i = \beta$$

i.e., don’t oversaturate

$$0 = x_i = \text{OPT}_i \quad \text{all } i$$

- **Intuition:** Strategy should look more like OPT than Nash

\[\text{OPT} \quad \text{Nash}\]
Reduction (con’d)

Partition: given n numbers $a_1,\ldots,a_n$ summing to $A$, is there a subset with sum $A/2$?

- amount of controlled jobs = $A/2$
- saturate machine $i \iff$ pick $a_i$
- “yes” Instance $\Rightarrow$ P has value $A/4$
- “no” Instance $\Rightarrow$ P has value $< A/4$
Open Questions

Approximating the optimal strategy:

• LLF is best possible using OPT as a lower bound
• better guarantees for LLF via a better lower bound?
• more sophisticated algorithms?
  - existence of a (F)PTAS?

Also:

• min $\beta$ sufficient to recover OPT
  - see also [KLO 97]
General Graphs

Open:

• for general latency fns, fixed $\beta$: a strategy inducing an equilibrium w/ cost $= f(\beta) \times \text{opt}$
  - $1/\beta$ not achievable in general graphs (!)
  - maybe $2/\beta$? (or $O(n)$)

• for linear latency fns, a strategy w/ cost $< 4/3 \times \text{opt}$
  - e.g., is $8/7$ achievable for $\beta=\frac{1}{2}$?