Stackelberg Scheduling Strategies

Tim Roughgarden
Cornell University
The Model

- $m$ machines $1, 2, \ldots, m$
- A quantity $r$ 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

- which job assignments are "stable"?
  - jobs controlled by selfish, noncooperative agents
  - no job wants to switch machines (no job should be envious)

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

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
\hline
& & & & & & & & & & \\
\hline
& & & & & & & & & & \\
\hline
& & & & & & & & & & \\
\hline
& & & & & & & & & & \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}
\]
More on Equilibrium Assignments

\[
\begin{align*}
\frac{1}{2} & \quad \frac{1}{2} \\
\hline
l_1(x) = 1 & \quad l_2(x) = x^2 \\
\hline
0 & \quad 1
\end{align*}
\]

**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 \textit{system performance}
- can optimize in poly-time

Example:

\[
\begin{array}{c|c|c}
\frac{1}{2} & 1 & \text{cost} = \\
\frac{1}{2} & x^2 & \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{4} = ?
\end{array}
\]
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: \(r=1, k \text{ large}\)

\[
\begin{array}{c|c}
0 & 1 \\
\hline
1 & x^k
\end{array}
\quad \text{vs.} \quad
\begin{array}{c|c}
? & 1 \\
\hline
1 & x^k
\end{array}
\]

Nash costs 1  
OPT costs \(\approx 0\)
Near-Optimal Nash Assignments

Old Approach: weaken model, compare Nash vs. OPT (due to [Roughgarden/Tardos 00])

- general latency fns, weaker OPT

**Thm 1:** cost of Nash = cost of OPT at rate $2r$

- stronger OPT, linear latency fns ($l_i(x) = a_i x + b_i$)

**Thm 2:** cost of Nash = $4/3$ cost of OPT (at rate $r$)
Taming Selfishness through a Manager

New Approach:

• 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]
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

\[ \frac{1}{\beta} \times \text{cost of opt assignment} \]

(\( \beta = \) fraction of centrally assigned jobs)

**Fact:**

\( \frac{1}{\beta} \times \text{OPT} \) is best possible
Our Results - Linear Latency Functions

Theorem 2: Can efficiently compute a strategy inducing an equilibrium with cost

\[ \frac{4}{(3+\beta)} \times \text{cost of OPT} \]

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

Fact:

\[ \frac{4}{(3+\beta)} \times \frac{1}{1-\beta} \]

OPT is best possible
What makes a strategy (in)effective?

The Scale Strategy

- compute optimal assignment \( x \) of all jobs, assign centrally controlled jobs via \( \beta \cdot x \)

Moral: avoid machines that selfish users will (over)use anyways
The LLF Strategy

Largest Latency First (LLF):
- compute opt, x, for all jobs
- assign $x_i$ jobs to $i$ in order of decreasing $l_i(x_i)$'s (until no managed jobs remain)

$$
\begin{align*}
1 & \quad 2x & \quad (3/2)x \\
5/12 & \quad 1/4 & \quad 1/3 \\
\end{align*}
$$

**OPT ($r=1$)**

$$
\begin{align*}
1 & \quad 2x & \quad (3/2)x \\
5/12 & \quad 1/12 & \quad 0 \\
\end{align*}
$$

**LLF ($\beta = \frac{1}{2}$)**
LLF with General Latency Functions

**Theorem 1:** The LLF strategy induces an equilibrium with cost \( \frac{1}{\beta} \times \text{cost of opt assignment} \).

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

**Base case:** \( \beta \) common latency = \( L \)

- LLF \( \Rightarrow \) Machine 1’s latency \( \geq L \)
- OPT has \( \geq \beta \) jobs on machine 1
  \( \Rightarrow \) OPT pays \( \geq \beta L \), we pay \( L \)
LLF with Linear Latency Functions

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

Main difficulty:
• previous argument too weak for:

• need detailed study of Nash, OPT when all latencies are linear
Computing Optimal Strategies

**We've seen:** LLF has the best possible *worst-case* guarantee

**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:** Opt, Nash assignments
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?
  - Thm 3 is reduction from Partition
  - existence of a (F)PTAS?
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 $\beta=\frac{1}{2}$?