Lecture Notes on Congestion Games

Berthold Vöcking

Department of Computer Science
RWTH Aachen

SS 2005
Definition and Classification

Relationship to Potential Games

Complexity of congestion games
Description of Congestion Games

- $n$ agents share a set of resources $E$
- strategy space of player $i$ is $S_i \subseteq 2^E$
- latency for resource $e \in E$ is denoted by $\ell_e$ and depends on the number of players using $e$, that is, $\ell_e : \{1, \ldots, n\} \rightarrow \mathbb{N}$
- edge latencies are assumed to be non-decreasing
- each agent $i$ aims at choosing a strategy $S \in S_i$ such that her latency $\sum_{e \in S} \ell_e$ is as small as possible
Subclasses of Congestion Games

- **network congestion games**: the strategy space $S_i$ of player $i$ corresponds to the set of paths between a source $s_i$ and a destination $t_i$ in an underlying graph $G = (V, E)$

- **symmetric congestion games**: all players have the same strategy space

- **symmetric network congestion games**: all players have the same source and destination

- **single-choice congestion games**: each strategy consists only of a single resource like, e.g., routing on parallel links
Example for a symmetric network congestion game

```
  s --------- 1,2,8 --------- t
    |                  |
2,3,5             4,6,7
    |                  |
  s --------- 2,3,6 --------- t
    |                  |
2,3,5             2,3,6
  ```
Congestion games – pure equilibria

**Theorem:** (Rosenthal 1973)
Every congestion game admits a pure Nash equilibrium.
Rosenthal’s analysis

The theorem follows by a nice potential function argument.

- Let $s$ denote any state of the game, i.e., any allocation of pure strategies.
- Let $n(e, s)$ the number of agents that use resource $e$ in state $s$.
- The potential of $s$ is defined by

$$\phi(s) = \sum_{e \in E} \sum_{i=1}^{n(e, s)} \ell_e(i).$$
Lemma: Let \( s \) be any state. Suppose we go from \( s \) to a state \( s' \) by a strategy switch of a single player \( i \) that improves his latency by an amount of \( \delta \geq 0 \). Then \( \phi(s') = \phi(s) - \delta \).

**Proof:**

- The potential \( \phi(s) \) can be calculated by inserting the agents one after the other in any order, and summing the latencies of the players at the point of time at their insertion.
- W.l.o.g., agent \( i \) is the last player that we insert when calculating \( \phi(s) \). Then the potential accounted for agent \( i \) corresponds to the latency of player \( i \) in state \( s \).
- When going from \( s \) to \( s' \), the latency of \( i \) decreases by \( \delta \), and, hence, also the potential decreases by \( \delta \).
Illustration of Rosenthal’s potential function

The potential of the following "state" can be calculated by ...

\[ ... \]
Illustration of Rosenthal’s potential function

... adding first the red path ...

potential of red path = 2 + 2 = 4
Illustration of Rosenthal’s potential function

... adding then the green path ...

potential of green path = 2 + 4 = 6
Illustration of Rosenthal’s potential function

... and finally the blue path.

potential of blue path = 3 + 1 + 6 = 10
overall potential = 4 + 6 + 10 = 20
Now suppose the blue path improves her latency from 10 to 6.

Then the overall potential becomes $4 + 6 + 6 = 16$. 
Rosenthal’s analysis, continued

- Suppose we start in any state $s$ and iteratively decrease the potential by repeatedly applying local improvement steps of single agents.
- The lemma shows that every local improvement step decreases the potential at least by one unit.
- As the potential cannot drop below zero, we reach a pure Nash equilibrium (which corresponds to a local optimum of the potential function) after at most $\phi(s)$ steps.

Thus, the existence of pure Nash equilibria is shown.
Potential function directly implies a pseudopolynomial algorithm for computing a pure Nash equilibrium in general congestion games.

For which kind of congestion games can we compute pure Nash equilibria in polynomial time?
### Complexity of congestion games

<table>
<thead>
<tr>
<th></th>
<th>network games</th>
<th>general games</th>
</tr>
</thead>
<tbody>
<tr>
<td>symmetric</td>
<td>\exists \text{ poly-time Algo} *</td>
<td>PLS-complete</td>
</tr>
<tr>
<td>asymmetric</td>
<td>PLS-complete</td>
<td>PLS-complete *</td>
</tr>
</tbody>
</table>

[Fabrikant, Papadimitriou, Talwar 2004]

We will present the proofs for the marked entries.
Polynomial time algorithm for symmetric network congestion games

Transform the problem into min-cost flow problem:

- Each edge is replaced by $n$ parallel edges of capacity 1 each.
- The $i$th copy of edge $e$ has cost $\ell_e(i)$, $1 \leq i \leq n$.
- We seek for an (integer) flow of value $n$ from the source to the destination with minimum cost.

The optimal solution corresponds to a state of the congestion game. This state minimizes Rosenthal’s potential function and, hence, is a Nash equilibrium.
Illustration of the min-cost flow problem

The numbers refer to the cost of the individual edges. All capacities are set to 1.
The complexity class PLS (Polynomial Local Search)

Problems in PLS:
- optimization problem Π with neighborhood Γ
- for every solution s, Γ(s) denotes neighborhood of s
- given s, the neighborhood Γ(s) can be “evaluated efficiently”, that is, there is a poly-time algorithm
  - deciding in whether s is a local optimum wrt Γ, i.e., there is no better solution in Γ(s), and
  - returning a better solution than s from Γ(s) if s is not locally optimal.
- goal: find local optimum wrt Γ

The problem of finding a pure Nash equilibrium in congestion games is in PLS as it is equivalent of finding a local optimum of ϕ.
The complexity class PLS (Polynomial Local Search)

**PLS reductions:** Given two PLS problems $\Pi_1$ and $\Pi_2$ find a mapping from the instances of $\Pi_1$ to the instances of $\Pi_2$ such that

- the mapping can be computed in polynomial time,
- the local optima of $\Pi_1$ are mapped to local optima of $\Pi_2$, and
- a local optimum of $\Pi_1$ can be constructed from a local optimum in $\Pi_2$ in polynomial time.
NAE-SAT (Not-All-Equal-SAT)

- **Input:** A list of 3-clauses $c_1, \ldots, c_m$, $c_i = (x_i^1, x_i^2, x_i^3)$, only positive literals, each of which having a weight $w_i$
- **Objective:** Maximize the weighted number of clauses in which not all literals have the same value
- **Neighborhood:** Flips of single variables.

A technically complicated master reduction (that we will not present here) yields the following result.

**Theorem:** NAE-SAT is PLS-complete.
**Theorem:** Finding a Nash equilibrium in a general congestion game is PLS-complete.

**Proof:** PLS-Reduction from NAE-SAT:

- For each variable there is an agent.
- For each clause $c_i$ there are two resources $e_i(0)$ and $e_i(1)$.
- An agent whose variable occurs as a literal in $c_i$:
  - uses $e_i(0)$ if the literal in $c_i$ has value 0
  - uses $e_i(1)$ if the literal in $c_i$ has value 1
- The latency functions for $e_i(0)$ and $e_i(1)$ are of the form $(0, 0, w_i)$, that is, the latency is $w_i$ if all agents belonging to the clause use the resource and 0, otherwise.

Obviously the described mapping and its inverse can be computed in polynomial time. It remains to show that the mapping preserves the local optima.
Claim: There is a one-to-one mapping between the local optima of the two problems.

Proof: We focus on a single clause $c_i$.

- One can collect $w_i$ units from clause $c_i$ by flipping one of its variables if and only if either resource $e_i(0)$ or resource $e_i(1)$ was used by three agents.

- Thus, a flip increases the objective value of the NAE-problem (wrt $c_i$) by $w_i$ if and only if it also decreases the potential (wrt either $e_i(0)$ or $e_i(1)$) by the same amount.

Consequently, flips have the same effect on the objective functions of both problems (with inverted sign). This implies the claim and, thus, the theorem is shown.
Some open problems

- Find an efficient algorithm for computing a $(1 + \epsilon)$-approximate equilibrium, i.e., no agent can improve its latency by a factor of more than $(1 + \epsilon)$. 

- Show existence of a polynomial length local improvement path leading to a $(1 + \epsilon)$-approximate equilibrium. 

- Find a simple (randomized) process that quickly converges to a $(1 + \epsilon)$-approximate equilibrium.