Computational Game Theory

Vincenzo Bonifaci

June 8, 2010

3 Congestion and Potential Games

Recall that a finite *normal form game* Γ is given by

- a finite set $K = \{1, \ldots, k\}$ (the set of *players*);
- for each player $i = 1, \ldots, k$, a finite set Σ_i (the *strategy sets*);
- for each player $i = 1, \ldots, k$, a function $u_i : \Sigma_1 \times \ldots \times \Sigma_k \to \mathbb{R}$ (the *utility functions*).

An element $s \in \Sigma := \Sigma_1 \times \ldots \times \Sigma_k$ is called a *state* of the game. For a given $s \in \Sigma$ and $i = 1, \ldots, k$, the set

$$B_i(s) := \{s' \in \Sigma : s'_j = s_j \text{ for all } j \neq i\}$$

is called the *neighborhood* of s with respect to player i.

A *pure Nash equilibrium* of Γ is a state that is locally optimal with respect to any player's neighborhood; that is, a state s such that, for every $i \in K$ and for every $s' \in B_i(s)$, $u_i(s) \geq u_i(s')$.

Consider the *improvement dynamics graph* associated to the game, which is the graph with node set Σ and an arc (s, s') whenever $s' \in B_i(s)$ for some i such that $u_i(s') > u_i(s)$. A pure Nash equilibrium corresponds to a *sink* in the graph, that is, a node with no successors.

Congestion games. In a *congestion game*, strategies and payoffs are defined in terms of a finite set E of *resources* and in terms of *congestion functions* (also *delay* functions) $d : E \times \{1, \ldots, k\} \to \mathbb{R}$. Every player has to select one among different subsets of resources: $\Sigma_i \subseteq 2^E$ for every $i = \{1, \ldots, k\}$. Define $x_s(e) = |\{i : e \in s_i\}|$. Then the cost experienced by player i is defined as

$$\text{cost}_i(s) = -u_i(s) := \sum_{e \in s_i} d_e(x_s(e)).$$

Thus, each player chooses a subset of the resources and the cost he pays is the total congestion of the resources he is choosing. The congestion of a resource only depends on the number of players choosing that resource. An important subclass (*network congestion games*) is obtained when E is the set of edges of a network, and every Σ_i is the set of paths connecting two terminals (a_i, b_i). We assume the network to be directed. Figure 1 shows an example of a network congestion game.

In the example, $E = \{e_1, e_2, e_3, e_4, e_5\}$, and $\Sigma_1 = \Sigma_2 = \Sigma_3 = \{\{e_1, e_2\}, \{e_1, e_3, e_5\}, \{e_4, e_5\}\}$. The congestion functions are indicated along the edges, for example $d_{e_3}(3) = 8$. If player 1 plays $\{e_1, e_2\}$,

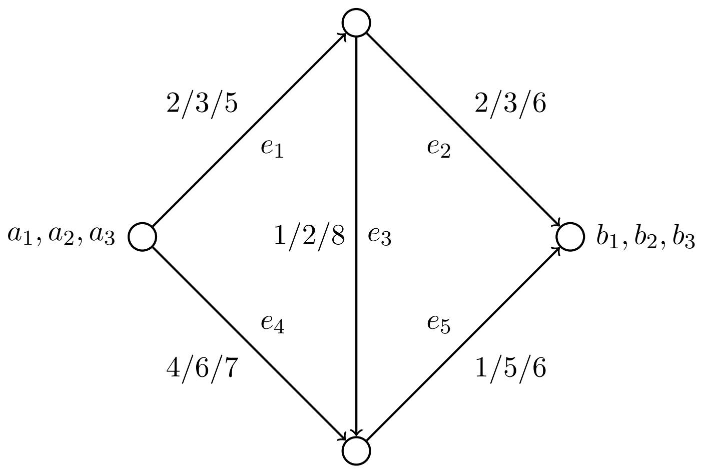

Figure 1: Example of a network congestion game.

player 2 plays $\{e_1, e_3, e_5\}$ and player 3 plays $\{e_4, e_5\}$, then the congestion level (x_e) on e_1 is 2, on e_2 is 1, on e_3 is 1, on e_4 is 1, and on e_5 is 2. So player 1 pays $3 + 2 = 5$, player 2 pays $3 + 1 + 5 = 9$, and player 3 pays $4 + 5 = 9$. This is not a Nash equilibrium, since for example player 2 can defect to $\{e_1, e_2\}$ and get a new cost of $3 + 3 = 6 < 9$.

Potential games. An *exact potential function* is a function $\phi : \Sigma \to \mathbb{R}$ such that

$$\phi(s) - \phi(s') = u_i(s') - u_i(s) \, (= \mathrm{cost}_i(s) - \mathrm{cost}_i(s'))$$

for each $s, s' \in \Sigma$ such that $s' \in B_i(s)$.

Example 3.1 (Prisoner's dilemma)**.** Assume the cost matrices are given by

$$\mathrm{cost}_1 = \begin{bmatrix} 4 & 1 \\ 5 & 2 \end{bmatrix}, \ \mathrm{cost}_2 = \begin{bmatrix} 4 & 5 \\ 1 & 2 \end{bmatrix},$$

The reader can verify that

$$\phi = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

is an exact potential function for the game.

A game admitting an exact potential function is called an *exact potential game.* The potential function always decreases along the edges of the improvement dynamics graph. So if Γ is a potential game, the improvement dynamics graph of Γ cannot have cycles. This means that one can always find a pure Nash equilibrium of a potential game using a local search algorithm: start at any state, and apply an improvement as long as possible. If the game is finite, the process must end and the final solution must be an equilibrium.

What is the relation between congestion and potential games?

Theorem 3.1 (Rosenthal 1973). *Every congestion game is an exact potential game.*

Let $\Gamma = (k, E, (\Sigma_i)_{i=1}^k, (d_e)_{e\in E})$ be a congestion game. We take

$$\phi(S) = \sum_{e\in E} \sum_{i=1}^{x_e(S)} d_e(i).$$

Proof. We only need to verify that ϕ is a potential function for Γ. Consider a deviation by any player. We can assume that this player is player number k; otherwise, we can renumber the players (the potential function is not affected by the numbering). Notice that we can write $\phi(S)$ as $\sum_{i=1}^k \phi_i(S)$, where, if we denote by $x_e^{\leq i}(S)$ the number of players with index $\leq i$ that use resource e in S,

$$\phi_i(S) = \sum_{e\in S_i} d_e(x_e^{\leq i}(S)).$$

We then obtain

$$\begin{aligned}
\phi(S) - \phi(S') &= \sum_{i=1}^k (\phi_i(S) - \phi_i(S')) \\
&= \phi_k(S) - \phi_k(S') \\
&= \sum_{e\in S_k} d_e(x_e^{\leq k}(S)) - \sum_{e\in S_k'} d_e(x_e^{\leq k}(S')) \\
&= \sum_{e\in S_k} d_e(x_e(S)) - \sum_{e\in S_k'} d_e(x_e(S')) \\
&= \mathrm{cost}_k(S) - \mathrm{cost}_k(S').
\end{aligned}$$

□

In fact, Monderer and Shapley proved that the converse of Rosenthal's theorem is true as well. We will not prove this result.

Theorem 3.2 (Monderer and Shapley 1996). *Every exact potential game is equivalent to a congestion game.*

Finding an equilibrium in a congestion game

A congestion game always has at least one pure Nash equilibrium, but how computationally hard is it to find one? It is always possible to apply the local search algorithm, however that might require a very large number of iterations to complete. Is there any faster way of finding the equilibrium? We show that the answer is "no", unless a whole host of local search problems can be solved efficiently. This seems unlikely, given the current state of research. The situation is akin to that of NP-complete problems; these problems are to be considered computationally hard unless some revolutionary development will occurr in the theory of algorithms. However the correct complexity class for reasoning about local search problems is not NP, but rather the class known as PLS.

Definition 3.2 (PLS)**.** A minimization problem $\Pi \in \mathsf{PLS}$ is given by:

1. a set of (polynomial-time recognizable) instances I over some fixed alphabet V;
2. for each instance $x \in I$, a set of *feasible solutions* $F_x \subseteq V^{p(|x|)}$ where $p(\cdot)$ is a fixed polynomial;
3. for each $x \in I$ and $y \in F_x$, a *neighborhood* $N_x(y) \subseteq F_x$;
4. a polynomial-time function **init** which, given $x \in I$, produces a feasible solution $\textbf{init}(x) \in F_x$;
5. a polynomial-time function **cost** which, on input $x \in I$ and $y \in V^{p(|x|)}$, determines whether $y \in F_x$ and, if so, computes a nonnegative integer $\text{cost}(x, y)$;
6. a polynomial-time function **improve** which, given $x \in I$ and $y \in F_x$ returns an $y' \in N_x(y)$ with $\text{cost}(x, y') < \text{cost}(x, y)$, or, if no such y' exists, returns **no**.

An instance of the PLS problem is: "Given $x \in I$, find a local minimum, that is, a $y \in F_x$ such that $\textbf{improve}(x, y)$=**no**." Maximization PLS problems can be defined analogously.

Every problem in PLS has a straightforward *standard search algorithm*: using function **init**, start with any feasible solution for the given instance, then call the function **improve** on this solution; if the response is **no** then we have found a local optimum. Otherwise the output of **improve** becomes the new current solution and we continue by invoking **improve** again, and so on. Since at every step the cost of the current solution has to decrease by at least one, sooner or later a local optimum has to be found. The problem with this algorithm is that the number of steps could be exponential in the size of the instance. Indeed, for every PLS-complete problem there are such instances. On the other hand, the number of steps can never be larger than the cost of the initial solution.

It should be now clear that, by the proof of Rosenthal's theorem, finding a pure Nash equilibrium for a congestion game is a problem in the class PLS: it is equivalent to finding a local optimum of the potential function ϕ, where the feasible solutions are all states and the neighborhood of a state is the set of states which arise from the deviation of just one player. However, this does not directly imply a polynomial algorithm, since, again, improvements of ϕ can be small and exponentially many.

We can extend the usual notion of reduction to problems in PLS.

Definition 3.3. A local search problem Π is PLS-*reducible* to another problem Π' if there are two polynomial-time algorithms A_1, A_2 with the following properties:

1. Algorithm A_1 maps an instance x of Π into an instance $A_1(x)$ of Π';
2. Algorithm A_2 maps an instance x of Π *together with* a solution y' of $A_1(x)$ into a solution y of x;
3. Whenever y' is a local optimum for $A_1(x)$, $A_2(x, y')$ is a local optimum for x.

Proposition 3.3. *If Π is PLS-reducible to Π', and Π' is solvable in polynomial time, then Π is solvable in polynomial time.*

Proposition 3.4. *If Π is PLS-reducible to Π', and $\Pi' \in \mathsf{PLS}$, then $\Pi \in \mathsf{PLS}$.*

Definition 3.4. A local search problem $\Pi \in \mathsf{PLS}$ is PLS-*complete* if each problem $\Pi' \in \mathsf{PLS}$ is PLS-reducible to Π.

Corollary 3.5. *If* Π *is* PLS*-complete and* Π *can be solved in polynomial time, then any problem in* PLS *can be solved in polynomial time.*

We now show that one cannot find a pure Nash equilibrium in a congestion game in polynomial time unless all PLS problems can be solved in polynomial time.

Theorem 3.6. *It is* PLS*-complete to find a pure Nash equilibrium in a congestion game.*

Proof. Let NOT-ALL-EQUAL 3SAT (NAE-3SAT) be the problem of finding, given a set of clauses, a truth assignment such that the truth values of the three literals in every clause are never all equal. Such a truth assignment is said to satisfy the clauses. A local search version of this problem is the following: given an instance of NAE-3SAT with weights on its clauses and containing positive literals only, find a truth assignment that satisfies a set of clauses whose total weight cannot be increased by flipping a variable. This problem, known as POSNAE-3SAT, was shown to be PLS-complete by Schäffer and Yannakakis in 1991.

Example 3.5. Consider the POSNAE-3SAT instance with four variables (y_1, y_2, y_3, y_4) and the clauses:

$$c_1 = (y_1, y_2, y_3); \qquad c_2 = (y_1, y_2, y_4); \qquad c_3 = (y_1, y_3, y_4); \qquad c_4 = (y_2, y_3, y_4)$$

with weights

$$w_1 = 1; \qquad w_2 = 2; \qquad w_3 = 2; \qquad w_4 = 3.$$

If we start with the solution $y = (\texttt{true}, \texttt{true}, \texttt{true}, \texttt{true})$, no clause is satisfied and the total weight is 0. So let's flip for example y_2 to `false`. The new total weight is $1 + 2 + 3 = 6$ because we satisfy clauses 1, 2, and 4. We can improve again by flipping y_4 to `false`. Now all clauses are satisfied so we reached a local optimum (in this case it is also a global optimum, but that is not important).

We now describe a PLS-reduction from POSNAE-3SAT to the problem of finding a pure equilibrium in a congestion game. Given an instance of POSNAE-3SAT, we construct a congestion game as follows: for each 3-clause c of weight w we have two resources e_c and e'_c, with congestion cost that is 0 if there are two or fewer players, and w otherwise. The players correspond to variables. Every player x has two strategies: one that contains all e_c's for clauses that contain x, and another that contains all e'_c's for the same clauses. Smaller clauses are implemented in a similar way.

Why did we use this construction? Because in any state s, the total cost in the game is exactly the sum of all clauses' weights, minus the value of the POSNAE-3SAT solution corresponding to s.

Example 3.6. The instance in the example above is reduced to a congestion game with four players and eight resources: $e_1, e'_1, \ldots, e_4, e'_4$. The strategy sets are:

$$\begin{aligned}
\Sigma_1 &= \{\{e_1, e_2, e_3\}, \{e'_1, e'_2, e'_3\}\}, \\
\Sigma_2 &= \{\{e_1, e_2, e_4\}, \{e'_1, e'_2, e'_4\}\}, \\
\Sigma_3 &= \{\{e_1, e_3, e_4\}, \{e'_1, e'_3, e'_4\}\}, \\
\Sigma_4 &= \{\{e_2, e_3, e_4\}, \{e'_2, e'_3, e'_4\}\}.
\end{aligned}$$

Moreover, $d_{e_1} = d_{e'_1} = (0,0,0,1,1)$, $d_{e_2} = d_{e'_2} = (0,0,0,2,2)$, $d_{e_3} = d_{e'_3} = (0,0,0,2,2)$, $d_{e_4} = d_{e'_4} = (0,0,0,3,3)$. In the solution $y = (\texttt{true}, \texttt{true}, \texttt{true}, \texttt{true})$, the congestion levels are:

$$\begin{array}{llll} d_{e_1}(3) = 1, & d_{e_2}(3) = 2, & d_{e_3}(3) = 2, & d_{e_4}(3) = 3, \\ d_{e'_1}(0) = 0, & d_{e'_2}(0) = 0, & d_{e'_3}(0) = 0, & d_{e'_4}(0) = 0. \end{array}$$

In this case, if player 2 flips his strategy to $\{e'_1, e'_2, e'_4\}$, the new costs become

$$\begin{array}{llll} d_{e_1}(2) = 0, & d_{e_2}(2) = 0, & d_{e_3}(3) = 2, & d_{e_4}(2) = 0, \\ d_{e'_1}(1) = 0, & d_{e'_2}(1) = 0, & d_{e'_3}(0) = 0, & d_{e'_4}(1) = 0. \end{array}$$

The total cost improved by exactly 1+2+3=6, which is the same improvement we have in the POSNAE-3SAT instance for flipping y_2.

Now, the pure Nash equilibria of the congestion game are exactly the states in which no player can improve his payoff by changing strategy. This means that in the corresponding truth assignment, no variable can be flipped in order to increase the weight of the clauses which are satisfied by that variable. Thus, any Nash equilibrium of the congestion game corresponds to a local optimum of the POSNAE-3SAT instance, which proves the theorem.

□