On approximate pure Nash equilibria in congestion games

Angelo Fanelli

1. **WCG**
2. **PNE**
   - Existence
   - Computation & Complexity (unweighted)
     - Matroid CG
3. **Approximate PNE**
   - Existence
   - Computation & Complexity (unweighted)
     - Asymmetric
     - Symmetric

Weighted Congestion Games (WCG)

- $N = \{1, 2, \ldots, n\}$, set of n players
- $E = \{e_1, e_2, \ldots, e_m\}$, set of m resources
- w_u, weight of player u
- $\Sigma_u \subseteq 2^E$, set of strategies of player u

A state of the game is given by an assignment of strategies to players

$$S = (s_1, s_2, \ldots, s_n) \qquad s_u \in \Sigma_u$$

$$\Sigma = \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$$

WCG

- $f_e : \mathbb{R}^+ \mapsto \mathbb{R}$, latency function of resource $e \in E$
  - $f_e(n_e(S))$, latency of e in state S
  - $n_e(S) = \sum_{u:e \in s_u} w_u$, congestion of e in state S
- $c_u(S) = w_u \sum_{e \in s_u} f_e(n_e(S))$, cost incurred by player u

Subclasses (players)

- [**Unweighted**] congestion games (CG)
  - $w_u = 1$, for every $u \in N$
  - $n_e(S) = \#$ of players using e in state S
  - $c_u(S) = \sum_{e \in s_u} f_e(n_e(S))$

Subclasses (strategy spaces)

- **Network** congestion games
  - $G = (V, E)$
  - $(s_u, t_u) \in V^2$, source-destination of player $u \in N$
  - $E = \{e_1, e_2, \ldots, e_m\}$, set of links
  - $\Sigma_u \subseteq 2^E$, set of paths of player $u \in N$ connecting s_u to t_u
- **Symmetric** congestion games
  - $\Sigma_u = \Sigma_w$, for every $u, w \in N$
- **Singleton** congestion games
  - $|s| = 1$, for every $s \in \Sigma_u$ and $u \in N$

Subclasses (latency functions)

- **linear** congestion games

$$f_e(x) = a_{e,1}x + a_{e,2}$$

- **polynomial** congestion games of degree $d \geq 1$

$$f_e(x) = a_{e,d}x^d + \ldots + a_{e,2}x^2 + a_{e,1}x + a_{e,0} = \sum_{i=0}^{d} a_{e,i}x^i$$

Size of the game

Number of bits required to represents the

- matrix of coefficients - $(a_{e,k})_{e \in E, k \in [1 \ldots d]}$
  - $O\Big((d+1) \cdot m \cdot \log(\max_{e,k} a_{e,k})\Big)$ bits
- vector of weights - $(w_u)_{u \in N}$
  - $O\Big(n \cdot \log(\max_{u \in N} w_u)\Big)$ bits
- vector of strategy sets - $(\Sigma_u)_{u \in N}$
  - $O\Big(n \cdot m \cdot \max_{u \in N} |\Sigma_u|\Big)$ bits
  - compact representation of strategy sets for networks

Terminology & Notation

- We use network terminology (paths, links, ...)
- $S = (s_1, s_2, \ldots, s_u, \ldots, s_n)$
  If player u deviates from s_u to s'_u, the new resulting state is

$$S' = (S_{-u}, s'_u) = (s_1, s_2, \ldots, \mathbf{s'_u}, \ldots, s_n)$$

Pure Nash equilibrium (PNE)

[Improvement] move

The deviation of a player to any path that strictly decreases his cost, e.g.,

$$c_u(S_{-u}, s'_u) < c_u(S)$$

Best-response move

The deviation of a player to the shortest path, e.g.,

$$c_u(S_{-u}, s'_u) \leq c_u(S_{-u}, \bar{s}_u) \qquad \forall \bar{s}_u \in \Sigma_u$$

PNE

Improvement (best-response) dynamics

A finite sequence of improvement (best-response) moves

Pure Nash equilibrium (PNE)

State in which no player can unilaterally perform an improvement move

Existence of PNE in WCG

Harks, Klimm, MOR '12

Every instance of WCG with continuous latency functions admits a PNE if and only if the latencies are linear or exponential

Existence of PNE in CG

Rosenthal, 1973

Every instance of CG admits a PNE, and it can be computed by Algorithm 1

Algorithm 1

1. Start with any state S
2. **While** S is not a PNE **do**

   Let $u \in N$ and $s'_u \in \Sigma_u$, such that $c_u(S_{-u}, s'_u) < c_u(S)$
   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Correctness of Algorithm 1

- It follows by a potential function argument (Rosenthal's potential function)

$$\Phi : \Sigma \mapsto \mathbb{R}$$

$$\Phi(S) = \sum_{e \in E} \sum_{i=1}^{n_e(S)} f_e(i)$$

- Φ decreases at every iteration
  Let $S' = (S_{-u}, s_i')$ the resulting state of an improvement move of player i from s_u to s_u', then

$$c_u(S) - c_u(S') = \Phi(S) - \Phi(S')$$

- The algorithms terminates in a finite number of steps
  - Φ gets only a finite number of values because Σ is finite

Running Time of Algorithm 1

- Finite sequence of states of the improvement dynamics

$$S^0, S^1, \ldots, S^k, S^{k+1}, \ldots$$

$$\Phi(S^0) > \Phi(S^1) > \ldots > \Phi(S^k) > \Phi(S^{k+1}) > \ldots$$

- The number of states is $||\Sigma_1| \cdot |\Sigma_2| \cdot \ldots \cdot |\Sigma_n||$
- Algorithm 1 terminates in at most $||\Sigma_1| \cdot |\Sigma_2| \cdot \ldots \cdot |\Sigma_n||$ steps
- Exponentially large in the size of the game

Complexity of PNE in CG

Fabrikant, Papadimitriou and Talwar, STOC '04

Computing a PNE in CG is PLS-complete

The relationship to Local Search

- The potential function allows us to interpret the problem of computing a PNE as a Local Search Problems

Local Search Problem

A **Local Search Problem** Π is given by its set of instances $\mathcal{I}_\Pi$ and it is either a maximization or a minimization problem. For every instance $I \in \mathcal{I}_\Pi$ we are given

- a set of feasible solutions $\mathcal{F}(I)$
- an objective function $C : \mathcal{F}(I) \mapsto \mathbb{R}$
- for every $S \in \mathcal{F}(I)$, a neighborhood $\mathcal{N}(S, I) \subseteq \mathcal{F}(I)$

Given an instance $\mathcal{I}_\Pi$, the problem is to find a local optimal solution S. That is $C(S) \leq C(S')$ for all $S' \in \mathcal{N}(S, I)$ (for minimization)

Polynomial Local Search Problems (PLS)

A local search problem Π belongs to PLS if the following polynomial algorithms exist

- an algorithm A which computes for every instance $I \in \mathcal{I}_\Pi$ an initial feasible solution $S \in \mathcal{F}(I)$
- an algorithm B which computes for every instance $I \in \mathcal{I}_\Pi$ and every feasible solution $S \in \mathcal{F}(I)$ the objective value $c(S)$
- an algorithm C which determines for every instance $I \in \mathcal{I}_\Pi$ and every feasible solution $S \in \mathcal{F}(I)$ whether S is locally optimal or not and finds a better solution in the neighborhood of S in the latter case

PLS-reducible and PLS-complete

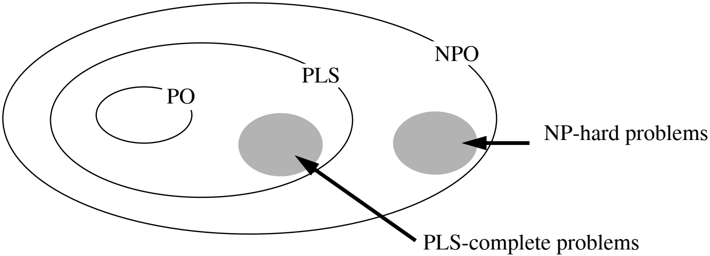
A problem Π_1 from PLS is PLS-reducible to Π_2 from PLS if there are polynomial computable functions f and g such that

- f maps instances $I \in \Pi_1$ to instances $f(I)$ of Π_2
- g maps pairs (S_2, I) with S_2 denoting a solution of $f(I)$ to solutions S_1 of I
- for all instances $I \in \Pi_1$, if S_2 is a local optimum of instance $f(I)$ then $g(S_2, I)$ is a local optimum of I

PLS-complete

A local seach problem Π from PLS is PLS-complete if every problem in PLS is PLS-reducible to Π

PLS-complete



Complexity of PNE in CG

Fabrikant, Papadimitriou and Talwar, STOC '04

Computing a PNE in CG is PLS-complete

It can be proved with a reduction from MAX-CUT with Flip-Neighborhood

MAX-CUT/Flip

- **Instance**: $G = (V, E)$ undirected with a weight $w_{\{i,j\}}$ for each $\{i, j\} \in E$
- **Feasible solution**: partition (A, B) of V
- **Objective function**: $\mathrm{Max} \quad U(A, B) = \sum_{\{i,j\}|i \in A, j \in B} w_{\{i,j\}}$;
- **Neighborhood function**: (A', B') is a neighbor of (A, B) iff it can be obtained from moving a single node from one side to the other one and $U(A, B) < U(A', B')$

Summary

Fabrikant, Papadimitriou and Talwar, STOC '04

	Network	General
Symmetric	P	PLS-complete
Asymmetric	PLS-complete	PLS-complete

Tractable case

Achermann, Röglin and Vöcking, FOCS '06

For every instance of Matroid Congestion Games (MCG), a PNE can be computed in polynomial time in the size of the game

Matroid

Matroid

A matroid M is a pair (E, I), where E is a finite set and I is a collection of subsets of E, i.e, $I \subseteq 2^E$ (called independent sets) with the following properties:

- $\emptyset \in I$
- (hereditary property). For each $A' \subseteq A \in E$, if $A \in I$ then $A' \in I$
- (exchange property). If $A, B \in I$ and $|A| > |B|$ then there exists $a \in A \setminus B$ such that $B \cup \{a\} \in I$

- The elements of I are called independent sets
- A maximal independent set is called basis of M
- The size of a maximal independent set is called the rank of M (denoted by $rank(M)$)

Matroid Congestion Games (MCG)

Matroid Congestion Games (MCG)

We call a Congestion Game $\mathcal{C} = (N, E, (\Sigma_u)_{u \in N}, (f_e)_{e \in E}, (c_u)_{u \in N})$ a Matroid Congestion Game if for every $u \in N$, let $M_u = (E, I_u)$ with $I_u = \{I \subseteq S | S \in \Sigma_u\}$

- M_u is a matroid
- Σ_u is the set of bases of M_u

- $rank(\mathcal{C}) = \max_{u \in N} rank(M_u)$

Examples

- Singleton Congestion Games
  - $rank = 1$
- Spanning Tree Congestion Games
  - given a network G, the strategy set of each player is a subset of the set of spanning trees of G

Computing a PNE in MCG

Achermann, Röglin and Vöcking, FOCS '06

For every instance of Matroid Congestion Games (MCG), Algorithm 2 computes a PNE in polynomial time in the size of the game

Algorithm 2

1. Start with any state S
2. **While** S is not a pure NE **do**

   Let $u \in N$, *and let* $s'_u \in \Sigma_u$ *be a shortest path such that* $c_u(S_{-u}, s'_u) < c_u(S)$

   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Conclusions on PNE

- Existence

Harks, Klimm, MOR '12

Every instance of WCG with continuous latency functions admits a PNE if and only if the latencies are linear or exponential

Rosenthal, 1973

Every instance of CG admits a PNE

- Computation

Fabrikant, Papadimitriou and Talwar, STOC '04

Computing a PNE in CG is PLS-complete

Some tractable cases: (e.g.) MCG, Network symmetric CG

ρ-apx PNE

ρ-move

The deviation of a player to any path that strictly decreases his cost by at least a factor $\rho \geq 1$, e.g.,

$$c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$$

Notice

An improvement move is a ρ-move for $\rho = 1$

ρ-apx PNE

State in which no player can unilaterally perform a ρ-move

Notice

A PNE is a ρ-apx PNE for $\rho = 1$

Motivations and Goals

Motivations

- PNE does not always exist and it may be difficult to compute
- For sufficiently large values of ρ there always exists a ρ-apx PNE and it is easy to compute
- Games are approximation of the real world

Goals

- Find the smallest value of ρ which guarantees existence and efficient computation of a ρ-apx PNE

Existence of ρ-apx PNE in WCG

Caragiannis, Fanelli, (working paper)

For every instance of polynomial WCG with degree $d \geq 1$, every sequence of d-moves leads to a d-apx PNE

Every d-move decreases the following potential function

$$\Psi(S) = \sum_{e \in E} \left(\frac{d_e}{d_e + 1} \left(\sum_{u : e \in s_u} w_u \right)^{d_e + 1} + \frac{1}{d_e + 1} \sum_{u : e \in s_u} w_u^{d_e + 1} \right)$$

where d_e is the degree of f_e and $d = \max_{e \in E} d_e$

Computing ρ-apx PNE in CG

Caragiannis, Fanelli, Gravin, Skopalik, FOCS '11

For every instance of polynomial CG with constant degree d and non-negative coefficients, a **$(q\text{–}\mathbf{Stretch}(\Phi) + \epsilon')$-apx PNE** is computable in polynomial time in the size of the game and $1/\epsilon'$, for any $\epsilon' > 0$ and $q > 1$

Skopalik and Vöcking, STOC '08

Computing a ρ-apx NE for CG is PLS-complete, for any $\rho \geq 1$

Computing ρ-apx PNE in CG

q-Stretch of the Rosenthal's potential

- $\mathrm{Neq}(q) = \{S \mid S \text{ is a } q\text{-apx PNE}\}$
- $\Phi(S) = \sum_{e \in E} \sum_{i=0}^{n_e(S)} f_e(i)$

$$\mathrm{q\text{–}Stretch}(\Phi) = \max_{S \in \mathrm{Neq}(q)} \frac{\Phi(S)}{\Phi_{\min}}$$

Bounds on the q-Stretch

- **Linear** latencies: $\mathrm{q\text{–}Stretch}(\Phi) = \mathbf{2 + O(q-1)}$
- **Polynomial** latencies: $\mathrm{q\text{–}Stretch}(\Phi) = d^{O(d)}$, for $q \in [1, 2]$

Preliminary to the Algorithm

Notation

- $\mathcal{BR}_u(S)$, any shortest path of player u in state S

$$c_u(S_{-u}, \mathcal{BR}_u(S)) = \min_{\bar{s}_u \in \Sigma_u} c_u(S_{-u}, \bar{s}_u)$$

- $\mathcal{BR}_u(\emptyset)$, any shortest path of u when no other player is participating in the game

- Optimistic cost of player u

$$p_u = \sum_{e \in \mathcal{BR}_u(\emptyset)} f_e(1)$$

- Minimum and maximum optimistic cost

$$\mathcal{L}_{\min} = \min_{u \in N} p_u \quad \text{and} \quad \mathcal{L}_{\max} = \max_{u \in N} p_u$$

Preliminary to the Algorithm
Linear CG

Algorithm 3

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **While** S is not a ρ-apx PNE **do**

   Let $u \in N$ *and* $s'_u \in \Sigma_u$, *such that* $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Assumption

- $f_e(x) = x$, for every $e \in E$

Observation

Let $T = \frac{\mathcal{L}_{\max}}{\mathcal{L}_{\min}}$.

Algorithm 3 returns a ρ-apx PNE in at most $\frac{n^2 T}{(\rho - 1)}$ steps

Preliminary to the Algorithm
Linear CG

Observation

Let $T = \frac{\mathcal{L}_{\max}}{\mathcal{L}_{\min}}$.
Algorithm 3 returns a ρ-apx PNE in at most $\frac{n^2 T}{(\rho-1)}$ steps

sketch of proof.

1. Upper bound the potential of the initial state

$$\Phi(S^0) \leq n^2 T \mathcal{L}_{\min}$$

2. Lower bound the decrease of the potential at each step

$$\Phi(S^k) - \Phi(S^{k+1}) \geq \mathcal{L}_{\min}(\rho - 1)$$

3. Combining the two inequalities, we get that the total number of steps is

$$\frac{n^2 T \mathcal{L}_{\min}}{\mathcal{L}_{\min}(\rho - 1)} \leq \frac{n^2 T}{(\rho - 1)}$$

Preliminary to the Algorithm
Linear CG

1. **Upper bound the potential of the initial state**
   - Initial state

$$S^0 = (s_1, s_2, \dots, s_n) \quad \text{where} \quad s_u = \mathcal{BR}_u(\emptyset)$$

   - For each player u

$$c_u(S^0) \leq n \cdot p_u \leq n \cdot \mathcal{L}_{\max} = nT\mathcal{L}_{\min}$$

   each edge can be used by at most n players

   - The potential is at most the sum of players' costs

$$\Phi(S) = \sum_e \sum_{j=0}^{n_e(S)} f(j) \leq \sum_e \sum_{j=0}^{n_e(S)} f(n_e(S)) = \sum_{u \in N} c_u(S)$$

   - Thus

$$\Phi(S^0) \leq \sum_{u \in N} c_u(S^0) \leq n^2 T\mathcal{L}_{\min}$$

2 **Lower bound the decrease of the potential at each step**

- The algorithm computes a sequence of states

$$S^0, S^1, \ldots, S^k, S^{k+1}, \ldots$$

- At step k

$$c_u(S^{k+1}) \leq \frac{c_u(S^k)}{\rho}$$

thus

$$\Phi(S^k)-\Phi(S^{k+1}) = c_u(S^k)-c_u(S^{k+1}) \geq c_u(S^{k+1})(\rho-1) \geq \mathcal{L}_{\min}(\rho-1)$$

$\square$

Preliminary to the Algorithm
Linear CG

Algorithm 3

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **While** S is not a ρ-apx PNE **do**

   Let $u \in N$ *and* $s'_u \in \Sigma_u$, *such that* $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Observation

Let $T = \frac{\mathcal{L}_{\max}}{\mathcal{L}_{\min}}$.

Algorithm 3 returns a *rho*-apx PNE in at most $\frac{n^2 T}{(\rho-1)}$ steps

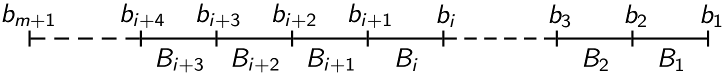
Preliminary to the Algorithm

- Players are statically classified into Blocks

$$B_m, B_{m-1}, \ldots, B_1$$

according to their optimistic cost

$$u \in B_i \quad \Leftrightarrow \quad p_u \in (b_{i+1}, b_i]$$



- $b_1 = \mathcal{L}_{\max}, \quad b_2 = \mathcal{L}_{\max}/g, \quad b_3 = \mathcal{L}_{\max}/g^2,$
  $\ldots \quad b_i = \mathcal{L}_{\max}/g^{(i-1)} \ldots$
  where g is a polynomial in n

- All players in the same block are polynomially related, i.e.,

$$\frac{b_i}{b_{i+1}} = g$$

- The number of blocks is polynomial in n

Preliminary to the Algorithm

Algorithm 3

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **While** S is not a ρ-apx PNE **do**

   Let $u \in N$ *and* $s'_u \in \Sigma_u$*, such that* $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Alg.3 runs Alg.4 sequentially on each block, from B_1 to B_m

Algorithm 4

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **For** $i = 1$ **to** m **do**
   1. **While** in S there exists a player u in B_i who has a ρ-move **do**

      Let $s'_u \in \Sigma_u$, such that $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

      $S \leftarrow (S_{-u}, s'_u)$
   2. **EndWhile**
3. **EndFor**

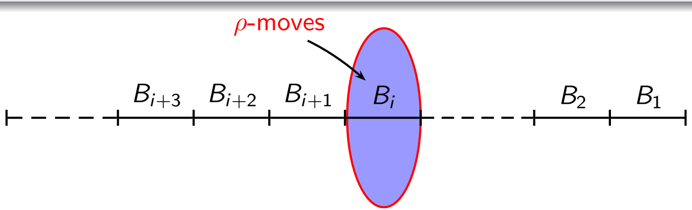
Preliminary to the Algorithm

Algorithm 4

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **For** $i = 1$ **to** m **do**
   1. **While** in S there exists a player u in B_i who has a ρ-move **do**

      Let $s'_u \in \Sigma_u$, such that $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

      $S \leftarrow (S_{-u}, s'_u)$
   2. **EndWhile**
3. **EndFor**



Phase i

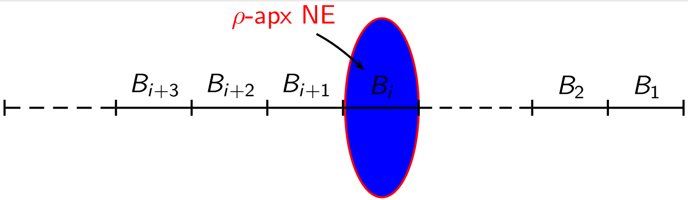
Preliminary to the Algorithm

Algorithm 4

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **For** $i = 1$ **to** m **do**
   1. **While** in S there exists a player u in B_i who has a ρ-move **do**

      Let $s'_u \in \Sigma_u$, such that $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

      $S \leftarrow (S_{-u}, s'_u)$
   2. **EndWhile**
3. **EndFor**



End of Phase i: Strategies in $B_1, B_2, \ldots, B_i$ irrevocably decided

Preliminary to the Algorithm

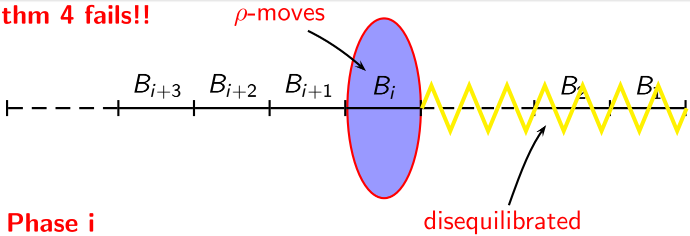
Algorithm 4

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$
2. **For** $i = 1$ **to** m **do**
   1. **While** in S there exists a player u in B_i who has a ρ-move **do**

      Let $s'_u \in \Sigma_u$, such that $c_u(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$

      $S \leftarrow (S_{-u}, s'_u)$
   2. **EndWhile**
3. **EndFor**

Algorithm 4 fails!!

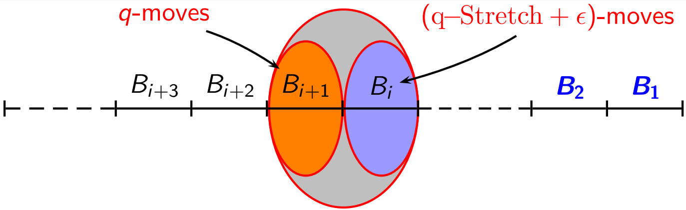


Phase i

Final Algorithm (for polynomial latencies)

Caragiannis, Fanelli, Gravin, Skopalik, FOCS '11

1. Let $S = (s_1, s_2, \dots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$, and $\underline{q \in (1, 2)}$
2. **For** $i = 1$ **to** $m - 1$ **do**
   1. **While** in S there exists a player u such that
      $u \in B_{i+1}$ and has a q-move **or**
      $u \in B_i$ and has a $(\text{q–Stretch} + \epsilon)$-move **do**
      $S \leftarrow (S_{-u}, \mathcal{BR}_u(S))$
   2. **EndWhile**
3. **EndFor**

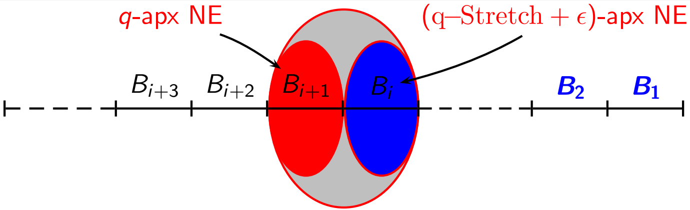


Phase i

Final Algorithm (for polynomial latencies)

Caragiannis, Fanelli, Gravin, Skopalik, FOCS '11

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$, and $\underline{q \in (1,2)}$
2. **For** $i = 1$ **to** $m - 1$ **do**
   1. **While** in S there exists a player u such that
      $u \in B_{i+1}$ and has a q-move **or**
      $u \in B_i$ and has a $(\text{q--Stretch} + \epsilon)$-move **do**
      $S \leftarrow (S_{-u}, \mathcal{BR}_u(S))$
   2. **EndWhile**
3. **EndFor**

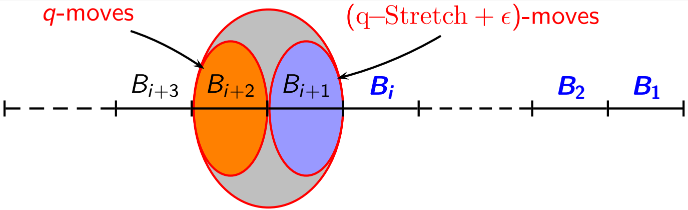


End of Phase i: Strategies in $B_1, B_2, \ldots, B_i$ irrevocably decided

Final Algorithm (for polynomial latencies)

Caragiannis, Fanelli, Gravin, Skopalik, FOCS '11

1. Let $S = (s_1, s_2, \ldots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$, and $\underline{q \in (1,2)}$
2. **For** $i = 1$ **to** $m-1$ **do**
    1. **While** in S there exists a player u such that
       $u \in B_{i+1}$ and has a q-move **or**
       $u \in B_i$ and has a $(\text{q–Stretch} + \epsilon)$-move **do**
       $S \leftarrow (S_{-u}, \mathcal{BR}_u(S))$
    2. **EndWhile**
3. **EndFor**

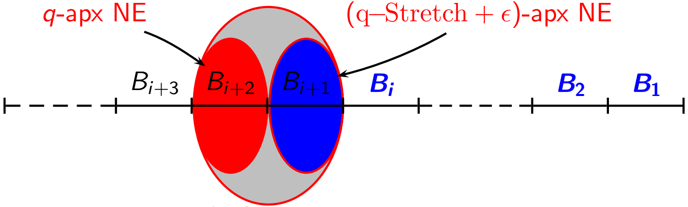


Phase i+1

Final Algorithm (for polynomial latencies)

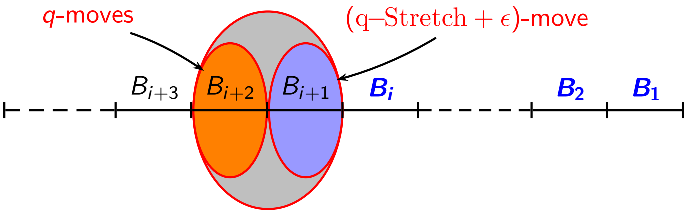
Caragiannis, Fanelli, Gravin, Skopalik, FOCS '11

1. Let $S = (s_1, s_2, \dots, s_n)$ such that $s_u = \mathcal{BR}_u(\emptyset)$, and $\underline{q \in (1,2)}$
2. **For** $i = 1$ **to** $m-1$ **do**
    1. **While** in S there exists a player u such that
       $u \in B_{i+1}$ and has a q-move **or**
       $u \in B_i$ and has a $(\text{q–Stretch} + \epsilon)$-move **do**
       $S \leftarrow (S_{-u}, \mathcal{BR}_u(S))$
    2. **EndWhile**
3. **EndFor**



End of Phase i+1: Strategies in $B_1, B_2, \dots, B_{i+1}$ irrevocably decide

Running time & Correctness

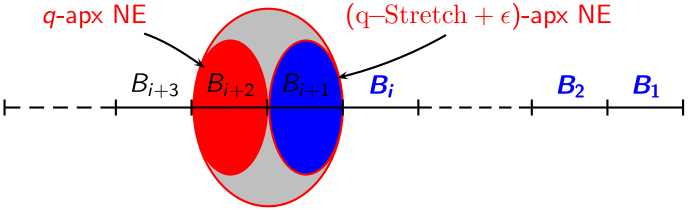


Phase i+1

Running time

- Polynomial number of phases
- Each phase runs in polynomial time

Running time & Correctness



End of Phase i+1

Running time

- Polynomial number of phases
- Each phase runs in polynomial time

Claim for phase $i+1$ (informally)

At the end of phase $i+1$, each player in $B_{i+1}, B_i, \ldots, B_1$ does not have a $(\mathbf{q\text{–}Stretch} + \boldsymbol{\epsilon'})$**-move**, where ϵ' is slightly larger than ϵ

Computing ρ-apx PNE in symmetric CG

Chen and Sinclair, SODA '06

Algorithms 5, on a symmetric CG with latencies satisfying the bounded jump condition, returns a ρ-apx NE, where $\rho = \frac{1}{1-\epsilon}$, in polynomial time in the size of the game and $1/\epsilon$, for any $\epsilon \in (0, 1)$

Algorithm 5

1. Start with any state S
2. **While** S is not a ρ-apx PNE **do**

   Let $u \in N$ *and* $s'_u \in \Sigma_u$, *such that* $c_u(S_{-u}, s'_u) < \frac{c_i(S)}{\rho}$

   $S \leftarrow (S_{-u}, s'_u)$
3. **EndWhile**

Computing ρ-apx PNE in symmetric CG

Bounded jump condition

A resource e satisfies the α-**bounded jump condition** if its latency function satisfies

$$f_e(t+1) \leq \alpha f_e(t)$$

for all $t \geq 1$ and α polynomially bounded in n

ρ-move is a symmetric CG

ρ-move

The deviation of a player to any path that decreases his cost by at least a factor $\rho \geq 1$,

$$c_i(S_{-u}, s'_u) < \frac{c_u(S)}{\rho}$$

- When $\rho = \frac{1}{1-\epsilon}$, with $\epsilon \in (0,1)$, we obtain that

$$c_u(S_{-u}, s'_u) < (1-\epsilon)c_u(S)$$

hence

$$c_u(S) - c_u(S_{-u}, s'_u) < \epsilon c_u(S)$$

and

$$\Phi(S) - \Phi(S_{-u}, s'_u) < \epsilon c_u(S)$$

Structure of the proof

- **Assumption**: In state S player u has cost $c_u(S) \geq \frac{\Phi(S)}{\beta}$ and makes an $\frac{1}{(1-\epsilon)}$-move leading to state S'
- This move must reduce c_u and hence Φ by more than $\frac{\epsilon \cdot \Phi(S)}{\beta}$

$$\Phi(S)(1 - \frac{\epsilon}{\beta}) \geq \Phi(S')$$

- Let S_{in} the initial state and S_ϵ the reached $\frac{1}{(1-\epsilon)}$-Nash equilibrium, applying recursively the previous argument for k steps, we get

$$\Phi(S_{in})(1 - \frac{\epsilon}{\beta})^k \geq \Phi(S_\epsilon)$$

- Assuming that Φ is a non-negative integer, then k is at most

$$k \leq \lceil \beta\epsilon^{-1} \log \Phi(S_{in}) \rceil \leq \lceil \beta\epsilon^{-1} \log \Phi_{\max} \rceil$$

Structure of the proof

- Number of steps
$$k \leq \lceil \beta \epsilon^{-1} \log \Phi_{\max} \rceil$$
- in order to be polynomial, β must be polynomial

- Main challange: Guarantee that at each step the cost of the moving player is $\geq \frac{\Phi(S)}{\beta}$ for polynomial values of β

Running time

Proof. (for restricted dynamic.)

Largest relative gain dynamic

In state S the move is made by a player u who maximize $\frac{c_u(S)-c_u(S_{-u},s'_u)}{c_u(S)}$

Lemma

If in state S, u is the moving player, then $c_u(S) \geq \frac{c_j(S)}{\alpha}$ for all $j \in N$

- Since $\Phi(S) \leq \sum_{j\in N} c_j(S)$, from Lemma we obtain $c_u(S) \geq \frac{\Phi(S)}{\alpha n}$
- By using the previous argument, we can choose $\beta = \alpha n$, and the number of moves is at most

$$k \leq \lceil \beta\epsilon^{-1} \log \Phi(S_{\max}) \rceil \leq \lceil \alpha n \epsilon^{-1} \log \Phi_{\max} \rceil$$

□

Running time

Lemma

If in state S, u is the moving player, then $c_u(S) \geq \frac{c_j(S)}{\alpha}$ for all $j \in N$

Proof.

- Player u moves from s_u to s'_u taking the game from S to $S' = (S_{-u}, s'_u)$
- Consider any player j and the resulting state if j, rather than u, had adopted s'_u. Let $S'' = (S_{-j}, s''_j = s'_u)$
- Since u moves and not j, then

$$\frac{c_j(S) - c_j(S'')}{c_j(S)} \leq \frac{c_u(S) - c_u(S')}{c_u(S)}$$

Running time

Proof. (cont.)

$$\frac{c_j(S) - c_j(S'')}{c_j(S)} \leq \frac{c_u(S) - c_u(S')}{c_u(S)}$$

- Let us compare $c_u(S')$ with $c_j(S'')$
- After u moves, since the latency of each resource e may be either $f_e(n_e(S))$ or $f_e(n_e(S)+1)$, and since $f_e(n_e(S)+1) \leq \alpha f_e(n_e(S))$ we get that, for each player j

$$c_j(S'') \leq \alpha c_u(S')$$

the claim follows combining the two inequalities $\square$

References

- Caragiannis, Fanelli, Gravin, Skopalik. *Approximate Pure Nash Equilibria in Weighted Congestion Games: Existence, Efficient Computation, and Structure*. ACM Trans. Economics and Comput. 2015
- Caragiannis, Fanelli, Gravin. *Short Sequences of Improvement Moves Lead to Approximate Equilibria in Constraint Satisfaction Games*. SAGT 2014