Algorithmic Game Theory
Final Exam

Juan Pablo Royo Sales
Universitat Politècnica de Catalunya

December 21, 2020

1 Exercise 1

A Network Congestion Game (NCG) is **symmetric** when $s_i = s$ and $t_i = t$, for $i \in N$ according to the definition that we have seen in class. Basically all the players, have the same source and destination endpoint s and t and all have the same path strategies.

The algorithm to find Pure Nash Equilibrium (PNE) in SNCG in *poly-time* is doing a reduction to Min-Cost Flow Problem (MINCOST) in order to find the Optimum of $\varphi(s)$.

Algorithm 1: Compute MINCOST reduction of SNCG

Input : $G = (V, E, s, t)$, d_e delay function
Output: MINCOST representation of the problem
for *each* $e \in E$ **do**
  for 1 *up to* d_e **do**
    $E \leftarrow E \cup \{e'_i\}$ such that $d_{e'_i} = 1$;
  $E \leftarrow E \setminus \{e\}$;
return N;

Basically the reduction is replacing each edge e with the amount of the delay function for e d_e of new edges but this time each of this edges with delay function equal to 1.

The MINCOST of the new network N minimizes $\varphi(s)$.

Since the algorithm is giving also a Local Optimum, then s is a PNE.

Based on this [FPT04]

2 Exercise 2

As we have seen in class the **Social Cost** is the sum of all players

$$C(s) = \sum_{u \in V} c_u(s) = \alpha|E| + \sum_{u,v \in V} d_G(u,v) \tag{1}$$

Every pair of vertices that is not *connected* its distances is ≥ 2, there is a lower bound for the social cost:

$$C(s) \geq \alpha|E| + 2|E| + 2(n(n-1) - 2|E|) \tag{2a}$$
$$= 2n(n-1) + (\alpha - 2)|E| \tag{2b}$$

2.1 (a)

In this case when $\alpha < 1$ the social Optimum is achieved when $|E|$ is maximum. So, every vertex is connected with each other. It is a complete graph. Any Nash Equilibrium (NE) should be of diameter 1 which implies that the complete graph is the only NE.

2.2 (b)

In the case $1 \leq \alpha < 2$, the social Optimum is achieved by a complete graph, but not NE. Any NE is for diameter at most 2. So, the social cost is equal to 2b and it is the worst cost when $|E|$ is minimum, which is $n - 1$ for a connected graph. Therefore the worst NE is the S_n Star.

2.3 (c)

As we have seen in class when $\alpha > 2$ we need to minimize the number of edges $|E|$ in order to reduce the cost, but at the same time the graph should be connected. Therefore only trees with diameter 2 have optimal cost. One example that has NE, with diameter 2 and minimum number of edges is a **Cycle Graph** with 4 vertex.

2.4 (d)

Since when $\alpha > n^2$ no player has any incentive to buy an edge, there is going to be the minimum amount possible of edges at the same time with the graph connected, that by definition are **Spanning Trees** as we have seen in class. All are NE because no one has incentive to change. The PoA is giving by:

$$PoA = \frac{c(T_n)}{c(S_n)} \tag{3a}$$
$$\leq \frac{\alpha(n-1) + (n-1)(n-1)}{\alpha(n-1) + 1 + 2n(n-1)} \tag{3b}$$
$$= O(1) \tag{3c}$$

Based on this [FLM+03].

3 Exercise 3

3.1 (a)

In order to show that it is a Simple Game (Simple Game), we need to show that the valuation function is **monotone**.

We assume in this game that

$$v(X) = \begin{cases} 1 \text{ } \textit{if } X \textit{ is a Dominant Set set of } G \\ 0 \text{ } \textit{otherwise} \end{cases}$$

And on the other hand we know that A Game is monotone if $v(C) \leq v(D)$ for any C, D such that $C \subseteq D$.

Lets take any $C, D \in G$ such that $C \subseteq D$ and lets consider the following cases:

- If D is a Dominant Set (Dominant Set), any subset of vertices on D, lets call C such that $C \subseteq D$, have 2 possibilities: either
  - C is also dominant because we remove some border vertex which adjacency vertices are connected to other $v \in C$, so $v(C) = v(D) = 1$

  - C is not dominant because selected subset is not a resulting Dominant Set, in which case $v(C) = 0 < v(D)$

- If D is non dominant then any subset is also non dominant so $v(C) = v(D) = 0$.

Then we can conclude that $v(C) \leq v(D)$, for any $C, D \in G, C \subseteq D$. So this is **monotone**.

Therefore it is a Simple Game.

3.2 (b)

Base on the theorem that states *A simple game has non-empty core if and only if it has a veto player*, we can check if there is a veto player for some *dominant game* in order to see if it has a non-empty core. An example of that is a S_n star in which the center vertex c belongs to any coalition that contains c but it cannot be remove from any of those coalition because if we remove it it is not a Dominant Set.

3.3 (c)

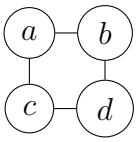
Lets analyze case by case. We are going to analyze IsStrong and IsProper

3.3.1 IsProper Analysis

A Game Γ is proper if $S \in W \implies N \setminus S \notin W$.

Let $S \in W$, then for every vertex $u \in S$, we have the following cases:

- $u \notin N \setminus S$, but that not implies that $N \setminus S \notin W$, because it can be another Dominant Set. For example if we have an cycle square connected in each vertex, and S are 2 adjacent vertices. $N \setminus S$ is still another W. In that case the game is not proper. Let see an example

  As we can see here for example if $\{a, b\} \in S$ which $\in W$, but the complement which is $\{b, d\} \in W$ as well.

- In order $N \setminus S \notin W$ we need that all $u \in W$ form a single Coalition. That means that S should be a Maximal Independent Set (Maximal

Independent Set), because by definition every Maximal Independent Set is a Dominant Set. In that case the Game is proper.

Therefore in order to analyze if the Game IsProper we need to provide an algorithm that check if the provided S is a Maximal Independent Set. Such an algorithm can be computed in *poly-time* because it is known that finding a Maximal Independent Set can be done in *poly-time* according to this [wik20]. Once we have the Maximal Independent Set of G, we need to check for each $u \in S$ if it is belongs to the Maximal Independent Set.

Algorithm 2: Compute if S IsProper

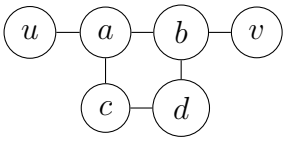
Input : $G = (V, E)$, S
Output: S IsProper
$MIS \leftarrow$ **Compute Maximal Independent Set in *poly-time***
for *each* $u \in S$ **do**
  if $u \notin MIS$ **then**
    return Not Proper;
return IsProper;

3.3.2 IsStrong Analysis

A Game Γ is strong if $S \notin W \implies N \setminus S \in W$.

Let $S \notin W$, then for every vertex $u \in S$, we have the following:

- $u \notin N \setminus S$, but that not implies that $N \setminus S \in W$, because it can be another subset that is $\notin W$. In that case the game is not strong. Let see an example



  As we can see here for example if $\{a, b, u\} \in S$ which $\notin W$, but the complement which is $\{b, d, v\} \notin W$ as well.

- In order $N \setminus S \in W$ we need that all $u \notin W$ are maximal losing form. That means that S should be a Minimal Vertex Cover (Minimal Vertex Cover), because by definition we know that the complement of Maximal Independent Set is a Minimal Vertex Cover. In that case the Game is proper.

It is well known that **decision Problem** for **Vertex Cover** is *NP-Complete*

for an arbitrary graph. So IsStrong is $NP-hard$ at least for deciding whether or not S is a Minimal Vertex Cover.

3.4 (d)

Given the fact as we have explained before we can calculate Maximal Independent Set (Maximal Independent Set) in *poly-time*, we can build a **Minimum Dominant Set** starting from the Maximal Independent Set and removing vertex checking that we still have an Dominant Set. But the issue with that is that we can have exponential number of possible ways of removing or combining the removal process from the Maximal Independent Set. So the problem is in *NP*, since we have different decision trees on the removal process.

4 Exercise 4

4.1 (a)

1. *A candidate a can win an election at node $u \in T$ $\iff$ a can win in 1 of u's children subelections and it can defeat one of the potential winner of other sibling child of u in a pairwise election.*

Proof.

Part 1. $\implies$
This part is trivial because of the structure of the Balance Tree.

If we take lets say this candidate a and this can win an election at node $u \in T$, we can think on that subtree which root is a that a is the winner of that subtree according to Cup definition.

Then if it is the winner it is because it should win in some sub-children pairwise election in order to appear at the top.

Since that a belongs to a subtree of T that subelections that wins should be for some of the subchildren of T.

Also it is trivial that a defeat any other sibling of u otherwise it shouldn't be at the root of this subtree.

Part 2. $\impliedby$
This part is not so trivial and we need to do it in two subparts.

1. *If a can win in 1 of u's children subelections of T*
2. *If a can defeat one of the potential winner of other sibling child of u in a pairwise election.*

If $1 \wedge 2 \implies$ *a can win an election at node $u \in T$.*

- Lets assume that point 1 holds, and a competes in a pairwise election with some b candidate in a subelection of T. If that happens and a wins to b the parent node of a and b would be a.
- At the same time in other sibling of u tree could be an election of in which there could be a potential winner lets say c. At some point that c are going to compete with a because they are going to raise to the tree roots. If that happens and as the problem state, if a can defeat this potential winner c, then a continue to the root and not c.

Therefore, a can win an election at node $u \in T$. ■

4.2 (b)

Since the voters in T should form a coalition in order to raise to the top to the tree the candidate $p \in A$, the idea of the algorithm is that this coalition manipulate elections in the leafs for those candidates that can be defeated by p in pairwise elections, constructing the pairwise competition in such a way that they can raise p to compete with the worst candidates. This basically can be done with a version of CSL algorithm [CSL07] but in a bottom up version instead of top down, and it takes $O(n^2)$

The algorithm is as following:

Algorithm 3: Compute CCM over $p \in A$

Input : $p \in A$ pref candidate, T Manipulator, C Binary Tree, D assignment of leaves
Output: True if p can win via manipulation, False otherwise

$MW \leftarrow$ConstructWinnersCCM(T,C,D) **if** $p \in MW$ **then**
    return True;
else
    return False;

Algorithm 4: ConstructWinnersCCM(T,C,D)

Input : T Manipulator, C Binary Tree, D assignment of leaves
Output: Winner Set Manipulated

if *isLeaf(C)* **then**
    return $\{C\}$;
else
    $W \leftarrow \{\}$; `// Possible Winner Set`
    leftTree $\leftarrow$ ConstructWinnersCCM(T, Left(C), D);
    rightTree $\leftarrow$ ConstructWinnersCCM(T, Right(C), D);
    forall $u \in leftTree$ **do**
        if $\exists v \in rightTree$ *such that* $(u, v) \in D \vee v \in T$ **then**
            $W \leftarrow \{u\}$
    forall $v \in rightTree$ **do**
        if $\exists u \in leftTree$ *such that* $(v, u) \in D \vee u \in T$ **then**
            $W \leftarrow \{v\}$
    return W;

Based on this [RW09].

References

[CSL07] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):14–es, June 2007.

[FLM+03] Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *Proceedings of the Twenty-Second Annual Symposium on Prin-*

ciples of Distributed Computing, PODC '03, page 347–351, New York, NY, USA, 2003. Association for Computing Machinery.

[FPT04] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 604–612, New York, NY, USA, 2004. Association for Computing Machinery.

[RW09] Tyrel Russell and Toby Walsh. Manipulating tournaments in cup and round robin competitions. In Francesca Rossi and Alexis Tsoukias, editors, *Algorithmic Decision Theory*, pages 26–37, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[wik20] wikipedia.org. Maximal independent set. `https://en.wikipedia.org/wiki/Maximal_independent_set#Complexity_class`, 2020. [Online; accessed 18-Dec-2020].