THE DISTRIBUTED COMPUTING COLUMN

BY

MARIO MAVRONICOLAS

Department of Computer Science, University of Cyprus
75 Kallipoleos St., CY-1678 Nicosia, Cyprus
mavronic@cs.ucy.ac.cy

A GAME ON A DISTRIBUTED NETWORK*

Vicky Papadopoulou

Department of Computer Science

University of Cyprus

P.O. Box 20537, Nicosia CY-1678, Cyprus

viki@cs.ucy.ac.cy

Abstract

Consider a distributed information network with harmful procedures called *attackers* (e.g., viruses); each attacker uses a probability distribution to choose a node of the network to damage. Opponent to the attackers is the *system protector* scanning and cleaning from attackers some part of the network (e.g., an edge or a simple path), which it chooses independently using another probability distribution. Each attacker wishes to maximize the probability of escaping its cleaning by the system protector; towards a conflicting objective, the system protector aims at maximizing the expected number of cleaned attackers. In [8, 9], we model this network scenario as a non-cooperative strategic game on graphs. We focus on two basic cases for the protector; where it may choose a single edge or a simple path of the network. The two games obtained are called as the *Path* and the *Edge* model, respectively. For these games, we are interested in the associated *Nash equilibria*, where no network entity can unilaterally improve its local objective. For the Edge model we obtain the following results:

*This work was partially supported by the IST Programs of the European Union under contract numbers IST-2001-33116 (FLAGS) and IST-2004-001907 (DELIS).

- No instance of the model possesses a pure Nash equilibrium.
- Every mixed Nash equilibrium enjoys a graph-theoretic structure, which enables a (typically exponential) algorithm to compute it.
- We coin a natural subclass of mixed Nash equilibria, which we call *matching Nash equilibria*, for this game on graphs. Matching Nash equilibria are defined using structural parameters of graphs
  - We derive a characterization of graphs possessing matching Nash equilibria. The characterization enables a linear time algorithm to compute a matching Nash equilibrium on any such graph.
  - Bipartite graphs and trees are shown to satisfy the characterization; we derive polynomial time algorithms that compute matching Nash equilibria on corresponding instances of the game.
- We proceed with other graph families. Utilizing graph-theoretic arguments and the characterization of mixed NE proved before, we compute, in polynomial time, mixed Nash equilibria on corresponding graph instances. The graph families considered are regular graphs, graphs with, polynomial time computable, r-regular factors and graphs with perfect matchings.
- We define the *social cost* of the game to be the expected number of attackers catch by the protector. We prove that the corresponding *Price of Anarchy* in any mixed Nash equilibria of the Edge model is upper and lower bounded by a linear function of the number of vertices of the graph.

Finally, we consider the more generalized variation of the problem considered, captured by the Path model. We prove that the problem of existence of a pure Nash equilibrium is $\mathcal{NP}$-complete for this model.

1 Introduction

Motivation and Framework. Although *Network Security* has been always considered to be a critical issue in networks, the recent huge growth of public networks (e.g. the Internet) made it even more very important [15]. This work considers a dimension of this area, related to the protection of a system from harmful entities (e.g. viruses, worms, trojan horses, eavesdroppers [4]). Consider an information network where the nodes of the network are insecure and vulnerable to infection by *attackers* such as, viruses, Trojan horses, eavesdroppers. In particular, at any time, a number of harmful entities is known (or an upper bound of this number) to be present in the network. A *protector*, i.e. system security software, is available in the system but it can guarantee security only to a limited

part of the network, such as a simple path or a single link of it, which it may choose using a probability distribution. Such limitations result from money and system performance costs caused in order to purchase a global security software or by the reduced efficiency or usability of a protected node. Each harmful entity targets a location (i.e. a node) of the network via a probability distribution; the node is damaged unless it is cleaned by the system security software. Apparently, the harmful entities and the system security software have conflicting objectives. The system security software seeks to protect the network as much as possible, while the harmful entities wish to avoid being caught by the software so that they be able to damage the network. Thus, the system security software seeks to maximize the expected number of viruses it catches, while each harmful entity seeks to maximize the probability it escapes from the system security software.

Naturally, we model this scenario as a non-cooperative multi-player strategic game played on a graph with two kinds of players: the *vertex players* representing the harmful entities, and the *edge* or the *path player* representing each one of the above two cases for the system security software considered; where it can choose a simple path or a single edge of the network, respectively. The corresponding games are called the *Path* and the *Edge* model, respectively. In both cases, the Individual Cost of each player is the quantity to be maximized by the corresponding entity. We are interested in the *Nash equilibria* [11, 12] associated with these games, where no player can unilaterally improve its Individual Cost by switching to a more advantageous probability distribution.

Summary of Results. Here we overview the most important results of [8, 9]. Our study is mainly focus on the Edge model where our results are summarized as follows:

- We prove that the model posses no pure Nash equilibrium (Theorem 3.1).
- We then proceed to study mixed Nash equilibria (mixed NE) of the Edge model. We provide a graph-theoretic characterization of mixed NE (Theorem 3.2). Roughly speaking, the characterization yields that the support of the edge player and the vertex players are an edge cover and a vertex cover of the graph and a subgraph of the graph, respectively. Given the supports, the characterization provides a system of equalities and inequalities to be satisfied by the probabilities of the players. Unfortunately, this characterization only implies an exponential time algorithm for the general case.
- We introduce *matching* Nash equilibria, which are a natural subclass of mixed Nash equilibria with a graph-theoretic definition (Definition 4.1). Roughly speaking, the supports of vertex players in a matching Nash equilibrium form together an independent set of the graph, while each vertex

in the supports of the vertex players is incident to only one edge from the support of the edge player.

- We provide a characterization of graphs admitting a *matching* Nash equilibrium (Theorem 4.4). We prove that a *matching* Nash equilibrium can be computed in linear time for any graph satisfying the characterization once a *suitable* independent set is given for the graph.

- We consider bipartite graphs for which we show that they satisfy the characterization of *matching* Nash equilibria; hence, they always have one (Theorem 5.4). More importantly, we prove that a *matching* Nash equilibrium can be computed in polynomial time for bipartite graphs (Theorem 5.5).

- Next, we proceed with other families of graphs. Combining the characterization of mixed Nash equilibria proved before with suitable graph-theoretic properties of each class addressed, we compute polynomial time mixed NE for each of them. These graph families include, trees, regular graphs, graphs that can be partitioned into vertex disjoint regular subgraphs, graphs with perfect matchings (Theorems 6.5, 6.6, 6.7, 6.9, respectively). Note that trees are also bipartite graphs. Thus, the algorithm for bipartite graphs can apply on them as well. However, the algorithm for trees provided, computes *matched* Nash equilibria in in significantly less time that the algorithm of bipartite graphs. This is achieved via suitable exploration of the special structure of a tree.

- We measure the system performance with respect to the problem considered utilizing the notion of the *social cost* [6]. Here, it is defined to be the number of attackers catch by the protector. We compute upper and lower bounds of the social cost in any mixed Nash equilibria of the Edge model. Using these bounds, we show that the corresponding Price of Anarchy is upper and lower bounded by a linear function of the number of vertices of the graph (Theorem 7.2).

Finally, we consider a more generalized case of the problem considered, represented by the Path model. We prove that the problem of existence of pure Nash equilibria in this model is $\mathcal{NP}$-complete (Theorem 8.2). This result opposes interestingly with the corresponding non-existence result of the Edge model, proved before and indicates some fascinating dimensions of the yet unexplored research area considered here.

Significance and Related Work. Our work joins the booming area of *Algorithmic Game Theory*. At the same time, it contributes in the subfield of *Network*

Security, related to the protection of a network from harmful entities (e.g. viruses, worms, malicious procedures, or eavesdroppers [4]). This work is the *first* work (with an exception of [2]) to model *network security problems* as strategic game and study its associated Nash equilibria. In particular, [2] is a part of a relevant research line related on *Interdependent Security* games [5]. In such a game, a large number of players must make individual investment decisions related to security, in which the ultimate safety of each participant may depend in a complex way on the actions of the entire population. Another related work is that of [4], studying the feasibility and computational complexity of two privacy tasks in distributed environments with *mobile eavesdroppers*; of distributed database maintenance and message transmission. A mobile eavesdropper is a computationally unbounded adversary that move its bugging equipment within the system.

This work is one of the only few works highlighting a fruitful interaction between *Game Theory* and *Graph Theory*. In [2], the authors consider inoculation strategies for victims of viruses and establishes connections with variants of the Graph Partition problem. In [1], the authors study a two-players game on a graph, establish connections with the k-server problem and provide an approximate solution for the simple network design problem.

Our results contribute towards answering the general question of Papadimitriou [14] about the complexity of Nash equilibria for our special game. We believe that our *matching* Nash equilibria (and/or extensions of them) will find further applications in other network games and establish themselves as a candidate Nash equilibrium for polynomial time computation in other settings as well.

2 Framework

Throughout, we consider an undirected graph $G(V, E)$, with $|V(G)| = n$ and $|E(G)| = m$. Given a set of vertices $X \subseteq V$, the graph $G \backslash X$ is obtained by removing from G all vertices of X and their incident edges. A graph H, is an *induced* subgraph of G, if $V(H) \subseteq V(G)$ and $(u, v) \in E(H)$, whenever $(u, v) \in E(G)$. For any vertex $v \in V(G)$, denote $Neigh(v) = \{u : (u, v) \in E(G)\}$, the set of neighboring vertices of v. For a set of vertices $X \subseteq V$, denote $Neigh(X) = \{u \notin X : (u, v) \in E(G)$ for some $v \in X\}$. Denote $\Delta(v) = |Neigh(v)|$ the degree of vertex v in G and $\Delta(G) = \max_{v \in V} |Neigh(v)|$ the maximum degree of G. A *simple* path, P, is a path of G with no repeated vertices, i.e. $P = \{v_1, \cdots, v_i \cdots, v_k\}$, where $1 \leq i \leq k \leq n$, $v_i \in V$, $(v_i, v_{i+1}) \in E(G)$ and each $v_i \in V$ appears at most once in P. Denote $\mathcal{P}(G)$ the set of all possible paths in G. For a tree graph T denote $root \in V$, the root of the tree and $leaves(T)$ the leaves of the tree T. For any $v \in V(T)$, denote $parent(v)$ the parent of v in the tree and $children(v)$ its children in the tree T. For any $A \subseteq V$, let $parents(A) := \{u \in V : u = parent(v),\ v \in A\}$. For all above properties of a

graph G, when there is no confusion, we omit G.

2.1 The model

Definition 2.1. *An information network is represented as an undirected graph $G(V, E)$. The vertices represent the network hosts and the edges represent the communication links. For $\mathsf{M} = \{\mathsf{P}, \mathsf{E}\}$, we define a non-cooperative game $\Pi_{\mathsf{M}}(G) = \langle \mathcal{N}, \{S_i\}_{i\in\mathcal{N}}, \{\mathsf{IC}\}_{i\in\mathcal{N}} \rangle$ as follows:*

- *The set of players is $\mathcal{N} = \mathcal{N}_{vp} \cup \mathcal{N}_p$, where $\mathcal{N}_{vp}$ is a finite set of* vertex *players vp_i, $i \geq 1$, $p = \{pp, ep\}$ and $\mathcal{N}_p$ is a singleton set of a player p which is either (i) the* path *player and $p = pp$ or (ii) the* edge *player and $p = ep$, in the case where $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

- *The strategy set S_i of each player vp_i, $i \in \mathcal{N}_{vp}$, is V; the strategy set S_p of the player p is either (i) the set of paths of G, $\mathcal{P}(G)$ or (ii) E, when $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively. Thus, the strategy set $\mathcal{S}$ of the game is $\left(\underset{i\in N_{vp}}{\times} S_i\right) \times S_p$ and equals to $|V|^{|\mathcal{N}_{vp}|} \times |\mathcal{P}(G)|$ or $|V|^{|\mathcal{N}_{vp}|} \times |E|$, when $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

- *Take any* strategy profile $\vec{\mathbf{s}} = \langle s_1, \ldots, s_{|\mathcal{N}_{vp}|}, s_p \rangle \in \mathcal{S}$, *also called a* configuration.

  - *The* Individual Cost *of vertex player vp_i is a function $\mathsf{IC}_i : \mathcal{S} \to \{0, 1\}$ such that $\mathsf{IC}_i(\vec{\mathbf{s}}) = \begin{cases} 0, & s_i \in s_p \\ 1, & s_i \notin s_p \end{cases}$; intuitively, vp_i receives 1 if it is not caught by the player p, and 0 otherwise.*
  - *The* Individual Cost *of the player p is a function $\mathsf{IC}_p : \mathcal{S} \to \mathbb{N}$ such that $\mathsf{IC}_p(\vec{\mathbf{s}}) = |\{s_i : s_i \in s_p\}|$.*

We call the games obtained as the Path *or the* Edge *model, for the case where $\mathsf{M} = \mathsf{P}$ or $\mathsf{M} = \mathsf{E}$, respectively.*

The configuration $\vec{\mathbf{s}}$ is a *pure Nash equilibrium* [11, 12] (abbreviated as *pure NE*) if for each player $i \in \mathcal{N}$, it maximizes IC_i over all configurations $\vec{\mathbf{t}}$ that differ from $\vec{\mathbf{s}}$ only with respect to the strategy of player i.

We consider *mixed strategies* for the Edge model. In the rest of the paper, unless explicitly mentioned, when referring to mixed strategies, these apply on the Edge model. A *mixed strategy* for player $i \in \mathcal{N}$ is a probability distribution over its strategy set S_i; thus, a mixed strategy for a vertex player (resp., edge player) is a probability distribution over vertices (resp., over edges) of G. A *mixed strategy profile* $\vec{\mathbf{s}}$ is a collection of mixed strategies, one for each player. Denote

$P_{\vec{s}}(ep, e)$ the probability that edge player ep chooses edge $e \in E(G)$ in $\vec{\mathbf{s}}$; denote $P_{\vec{s}}(vp_i, v)$ the probability that player vp_i chooses vertex $v \in V$ in $\vec{\mathbf{s}}$. Note $\sum_{v \in V} P_{\vec{s}}(vp_i, v) = 1$ for each player vp_i; similarly, $\sum_{e \in E} P_{\vec{s}}(ep, e) = 1$. Denote $P_{\vec{s}}(vp, v) = \sum_{i \in \mathcal{N}_{vp}} P_{\vec{s}}(vp_i, v)$ the probability that vertex v is chosen by some vertex player in $\vec{\mathbf{s}}$.

The *support* of a player $i \in \mathcal{N}$ in the configuration $\vec{\mathbf{s}}$, denoted $D_{\vec{s}}(i)$, is the set of pure strategies in its strategy set to which i assigns strictly positive probability in $\vec{\mathbf{s}}$. Denote $D_{\vec{s}}(vp) = \bigcup_{i \in \mathcal{N}_{vp}} D_{\vec{s}}(i)$; so, $D_{\vec{s}}(vp)$ contains all pure strategies (that is, vertices) to which some vertex player assigns strictly positive probability. Let also $ENeigh_{\vec{s}}(v) = \{(u, v) \in E : (u, v) \in D_{\vec{s}}(ep)\}$; that is $ENeigh_{\vec{s}}(v)$ contains all edges incident to v that are included in the support of the edge player in $\vec{\mathbf{s}}$. Given a mixed strategy profile $\vec{\mathbf{s}}$, we denote $(\vec{\mathbf{s}}_{-x}, [y])$ a configuration obtained by $\vec{\mathbf{s}}$, where all but player x play as in $\vec{\mathbf{s}}$ and player x plays the pure strategy y.

A mixed strategic profile $\vec{\mathbf{s}}$ induces an *Expected Individual Cost* IC_i for each player $i \in \mathcal{N}$, which is the expectation, according to $\vec{\mathbf{s}}$, of its corresponding Individual Cost (defined previously for pure strategy profiles). The mixed strategy profile $\vec{\mathbf{s}}$ is a *mixed Nash equilibrium* [11, 12] (abbreviated as mixed NE) if for each player $i \in \mathcal{N}$, it maximizes IC_i over all configurations $\vec{\mathbf{t}}$ that differ from $\vec{\mathbf{s}}$ only with respect to the mixed strategy of player i. We denote such a strategy profile as $\vec{s}^*$. Denote $BR_{\vec{s}}(x)$ the set of *best response (pure) strategies* of player x in a mixed strategy profile $\vec{\mathbf{s}}$, that is, $\mathsf{IC}_x(\vec{\mathbf{s}}_{-x}, y) \geq \mathsf{IC}_x(\vec{\mathbf{s}}_{-x}, y')$, $\forall\ y \in BR_{\vec{s}}(x)$ and $y' \notin BR_{\vec{s}}(x)$, $y' \in S_x$, where S_x is the strategy set of player x (see also [13], chapter 3). A *fully* mixed strategy profile is one in which each player plays with positive probability all strategies of its strategy set.

For the rest of this section, fix a mixed strategy profile $\vec{\mathbf{s}}$. For each vertex $v \in V$, denote $Hit(v)$ the event that the edge player hits vertex v. So, the probability (according to $\vec{\mathbf{s}}$) of $Hit(v)$ is $P_{\vec{s}}(Hit(v)) = \sum_{e \in ENeigh(v)} P_{\vec{s}}(ep, e)$. Define the minimum hitting probability $P_{\vec{s}}$ as $\min_v P_{\vec{s}}(Hit(v))$. For each vertex $v \in V$, denote $m_{\vec{s}}(v)$ the expected number of vertex players choosing v (according to $\vec{\mathbf{s}}$). For each edge $e = (u, v) \in E$, denote $m_{\vec{s}}(e)$ the expected number of vertex players choosing either u or v; so, $m_{\vec{s}}(e) = m_{\vec{s}}(u) + m_{\vec{s}}(v)$. It is easy to see that for each vertex $v \in V$, $m_{\vec{s}}(v) = \sum_{i \in \mathcal{N}_{vp}} P_{\vec{s}}(vp_i, v)$. Define the maximum expected number of vertex players choosing e in $\vec{\mathbf{s}}$ as $\max_e m_{\vec{s}}(e)$.

We proceed to calculate the Expected Individual Cost. Clearly, for the vertex player $vp_i \in \mathcal{N}_{vp}$,

$$\begin{aligned} \mathsf{IC}_i(\vec{\mathbf{s}}) &= \sum_{v \in V(G)} P_{\vec{s}}(vp_i, v) \cdot (1 - P_{\vec{s}}(Hit(v)) \\ &= \sum_{v \in V(G)} \left(P_{\vec{s}}(vp_i, v) \cdot (1 - \sum_{e \in ENeigh(v)} P_{\vec{s}}(ep, e) \right) \end{aligned} \tag{1}$$

For the edge player ep,

$$\begin{aligned}\mathsf{IC}_{ep}(\vec{\mathbf{s}}) &= \sum_{e=(u,v)\in E(G)} P_{\vec{\mathbf{s}}}(ep,e)\cdot(m_{\vec{\mathbf{s}}}(u)+m_{\vec{\mathbf{s}}}(v)) \\ &= \sum_{e=(u,v)\in E(G)}\left(P_{\vec{\mathbf{s}}}(ep,e)\cdot(\sum_{i\in N_{vp}} P_{\vec{\mathbf{s}}}(vp_i,u)+P_{\vec{\mathbf{s}}}(vp_i,v))\right) \qquad (2)\end{aligned}$$

Social Cost and Price of Anarchy. We utilize the notion of *social cost* [6] for evaluating the system performance related to the problem considered. A natural such measurement is the number of attackers catch by the system protector; a maximization of this quantity maximizes system's performance with respect to its safety from harmful entities. We therefore define,

Definition 2.2. *For model* M, $\mathsf{M}=\{\mathsf{P},\mathsf{E}\}$, *we define the* social cost *of configuration* $\vec{\mathbf{s}}$ *on instance* $\Pi_{\mathsf{M}}(G)$, $\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}})$, *to be the sum of vertex players of* $\Pi_{\mathsf{M}}(G)$ *arrested in* $\vec{\mathbf{s}}$. *That is,* $\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}}) = \mathsf{IC}_p(\vec{\mathbf{s}})$, *where* $p = pp$ *or* $p = ep$ *when* $\mathsf{M}=\mathsf{P}$ *or* $\mathsf{M}=\mathsf{E}$, *respectively. The system wishes to maximize the social cost.*

Definition 2.3. *For model* M, $\mathsf{M}=\{\mathsf{P},\mathsf{E}\}$, *we define the* price of anarchy, $\mathsf{r}(\mathsf{M})$ *to be,*

$$\mathsf{r}(\mathsf{M}) = \max_{\Pi_{\mathsf{M}}(G),\vec{s}^{\,*}} \frac{\max_{\vec{\mathbf{s}}\in\mathcal{S}} \mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{\mathbf{s}})}{\mathsf{SC}(\Pi_{\mathsf{M}}(G),\vec{s}^{\,*})}$$

2.2 Background from Graph Theory

Throughout this section, we consider the (undirected) graph $G=G(V,E)$.
$G(V,E)$ is *bipartite* if its vertex set V can be partitioned as $V=V_1\cup V_2$ such that each edge $e=(u,v)\in E$ has one of its vertices in V_1 and the other in V_2. Such a graph is often referred to as a V_1,V_2-bigraph. Fix a set of vertices $S\subseteq V$. The graph G is an S*-expander* if for every set $X\subseteq S$, $|X|\leq|Neigh_G(X)|$. For an integer r, graph G is r*-regular* if $\Delta(v)=r$, $\forall v\in V$.

A *factor* of a graph G is a sugraph $G_r\neq G$ such that $V(G_r)=V(G)$. An r*-regular factor* of G is a factor of it (not necessarily connected) which is also an r-regular graph. A *hamiltonian* path of a graph G is a simple path containing all vertices of G. A set $M\subseteq E$ is a *matching* of G if no two edges in M share a vertex. Given a matching M, say that set $S\subseteq V$ is *matched into* $V\backslash S$ in M if for every vertex $v\in S$, there is an edge $(v,u)\in M$ and $u\in V\backslash S$. A *vertex cover* of G is a set $V'\subseteq V$ such that for every edge $(u,v)\in E$ either $u\in V'$ or $v\in V'$. An *edge cover* of G is a set $E'\subseteq E$ such that for every vertex $v\in V$, there is an edge $(v,u)\in E'$. Say that an edge $(u,v)\in E$ (resp., a vertex $v\in V$) is *covered* by the

vertex cover V' (resp., the edge cover E') if either $u \in V'$ or $v \in V'$ (resp., if there is an edge $(u,v) \in E'$). A set $IS \subseteq V$ is an *independent set* of G if for all vertices $u, v \in IS$, $(u,v) \notin E$. Clearly, $IS \subseteq V$ is an independent set of G if and only if the set $VC = V \backslash IS$ is a vertex cover of G.

We will use the consequence of Hall's Theorem [3, Chapter 6] on the marriage problem.

Proposition 2.4 (Marriage's Theorem)**.** *A graph G has a matching M in which set $X \subseteq V$ is matched into $V \backslash X$ in M if and only if for each subset $S \subseteq X$, $|Neigh(S)| \geq |S|$.*

Note that the problem of finding a perfect matching of a graph (if there exists one) is equivalent to the problem of finding an 1-regular factor of the graph. The problem of finding a maximum matching of any graph can be solved in polynomial time [10]. Furthermore, a 2-regular factor of a graph (if there exists one) can be computed in polynomial time, via Tutte's reduction [16]; see also [7] for a survey in cycle covers problems of various sizes. By the above observations we get that there exists an exponential number of graphs that have polynomial time computable r-regular factors.

3 Nash Equilibria

All following sections, except the last one, are devoted to the Edge model. For pure Nash equilibria of the Edge model, in [8] we prove:

Theorem 3.1. *If G contains more than one edges, then $\Pi_E(G)$ has no pure Nash Equilibrium.*

Proof. Consider any graph G with at least two edges and any configuration $\vec{\mathbf{s}}$ of $\Pi_E(G)$. Let e the edge selected by the edge player in $\vec{\mathbf{s}}$. Since G contains more than one edges, there exists an $e' \in E$) not selected by the edge player in $\vec{\mathbf{s}}$, such that e and e' contain at least one different endpoint, assume u. If there is at least one vertex player located on e, it will prefer to alternate to u so that not to get arrested by the edge player and gain more. Thus, this case can not be a pure NE. Otherwise, no vertex player is located on edge e. This implies an individual cost of 0 for the edge player which the player can unilaterally improve by selecting any edge containing at least one vertex player. Thus, this case also can not be a pure NE for the instance, concluding that $\vec{\mathbf{s}}$ is not a pure NE. ■

Characterization of Mixed Nash Equilibria. Next we present a characterization of mixed Nash equilibria of the Edge model, proved in [8].

Theorem 3.2. *(***Characterization of Mixed NE***) A mixed strategy profile* $\vec{\mathbf{s}}$ *is a Nash equilibrium for any* $\Pi_{\mathsf{E}}(G)$ *if and only if:*

1. $D_{\vec{\mathbf{s}}}(ep)$ *is an* edge cover *of* G *and* $D_{\vec{\mathbf{s}}}(vp)$ *is a* vertex cover *of the graph obtained by* $D_{\vec{\mathbf{s}}}(ep)$.

2. *The probability distribution of the edge player over* E*, is such that, (a)* $P_{\vec{\mathbf{s}}}(Hit(v)) = P_{\vec{\mathbf{s}}}(Hit(u)) = \min_v P_{\vec{\mathbf{s}}}(Hit(v))$, $\forall\ u, v \in D_{\vec{\mathbf{s}}}(vp)$ *and (b)* $\sum_{e \in D_{\vec{\mathbf{s}}}(ep)} P_{\vec{\mathbf{s}}}(ep, e) = 1$.

3. *The probability distributions of the vertex players over* V *are such that, (a)* $m_{\vec{\mathbf{s}}}(e_1) = m_{\vec{\mathbf{s}}}(e_2) = \max_e m_{\vec{\mathbf{s}}}(e)$, $\forall\ e_1, e_2 \in D_{\vec{\mathbf{s}}}(ep)$ *and*
(b) $\sum_{v \in V(D_{\vec{\mathbf{s}}}(ep))} m_{\vec{\mathbf{s}}}(v) = \nu$.

Remark 3.3. *Note that the characterization does not implies a polynomial time algorithm for computing a mixed Nash equilibrium, since it involves solving a mixed integer programming problem.*

In [9], we also provide an estimation on the payoffs of the vertex players in any Nash equilibrium of the Edge model.

Claim 3.4. *For any* $\Pi_{\mathsf{E}}(G)$*, a mixed NE,* $\vec{s}^*$*, satisfies* $\mathsf{IC}_i(\vec{s}^*) = \mathsf{IC}_j(\vec{s}^*)$ *and* $1 - \frac{2}{|D_{\vec{s}^*}(vp)|} \leq \mathsf{IC}_i(\vec{s}^*) \leq 1 - \frac{1}{|D_{\vec{s}^*}(vp)|}$, $\forall i, j \in \mathcal{N}_{vp}$.

4 Matching Nash Equilibria

In [8] we introduce a family of configurations of the Edge model, called *matching*. Such configurations are shown to lead to mixed NE, called *matching* mixed NE. First, we provide a characterization for the existence of a *matching* mixed NE, shown in [8]. Using this characterization, we provide a polynomial time algorithm for the computation of *matching* Nash equilibria for any instance $\Pi_{\mathsf{E}}(G)$ of the problem, where the graph G satisfies the characterization. We remark applicability of the algorithm for a quite broad family of graphs, that of *bipartite graphs* (section 5).

Intuition behind Matching Nash equilibria. The obvious difficulty of solving the system of Theorem 3.2 directs us in trying to investigate the existence of some polynomially computable solutions of the system, corresponding to mixed NE of the game. To which configuration should we consider as *easy to compute*, we utilized the following way of thinking. A first observation is that finding a configuration that satisfies condition **2** of Theorem 3.2 seems the most difficult constrain (among the three conditions) to be fulfilled. This is so because it contains the

largest number of variables ($P_{\vec{s}}(ep, e)$, $\forall\ e \in E$) among the three conditions and each equation of it might involve up to $\Delta(G)$ such variables. Thus, let us consider the subtask of the system of computing function $P_{\vec{s}}(\cdot)$, $\forall e \in E$. Consider the case where the equations of condition **2.(a)** are *independent*, that is for each variable e, $P_{\vec{s}}(ep, e)$ appears in only one equation of condition **2.(a)**. Obviously, in this case the task becomes less difficult. Note that in such case, $D_{\vec{s}}(vp)$ constitutes an independent set of G. Moreover, when furthermore, each vertex of $D_{\vec{s}}(vp)$ is incident only in one edge of $D_{\vec{s}}(ep)$, then each equation of condition **2.(a)** contains only one variable, making the satisfaction of the condition even less difficult. Based on these thoughts, in [8], we define the following family of configurations which, as we show, can lead to mixed NE for the game. In the sequel, we investigate their existence and their polynomial time computation.

Definition 4.1. *A* **matching configuration** $\vec{s}$ *of* $\Pi_E(G)$ *satisfies:* **(1)** $D_{\vec{s}}(vp)$ *is an independent set of G and* **(2)** *each vertex v of* $D_{\vec{s}}(vp)$ *is incident to only one edge of* $D_{\vec{s}}(ep)$.

Claim 4.2. *[8] For any graph G, if in* $\Pi_E(G)$ *there exists a* matching *configuration which additionally satisfies condition* **1** *of Theorem 3.2, then there exists probability distributions for the vertex players and the edge player such that the resulting configuration is a mixed Nash equilibrium for* $\Pi_E(G)$. *These distributions can be computed in polynomial time.*

In the proof of the Claim, in [8], we consider any configuration $\vec{s}$ as stated by the Claim (assuming that there exists one) and the following probability distributions of the vertex players and the edge player on $\vec{s}$:

$$\begin{array}{l} \forall e \in D_{\vec{s}}(ep),\ P_{\vec{s}}(ep, e) := 1/|D_{\vec{s}}(ep)|, \\ \forall e' \in E,\ e' \notin D_{\vec{s}}(ep),\ P_{\vec{s}}(ep, e') := 0 \end{array} \tag{3}$$

$$\begin{array}{l} \forall\ i \in \mathcal{N}_{vp},\ \forall\ v \in D_{\vec{s}}(vp),\ P_{\vec{s}}(vp_i, v) := \frac{1}{|D_{\vec{s}}(vp)|}, \\ \forall u \in V,\ u \notin D_{\vec{s}}(vp),\ P_{\vec{s}}(vp_i, u) := 0 \end{array} \tag{4}$$

Then, it is shown that $\vec{s}$ satisfies all conditions of Theorem 3.2, thus it is a mixed NE.

Definition 4.3. *A* matching *configuration which additionally satisfies condition* **1** *of Theorem 3.2 is called a* **matching mixed NE**.

Furthermore, in [8], we characterize graphs that admit *matching* Nash equilibria.

Theorem 4.4. *For any graph G,* $\Pi_E(G)$ *contains a* matching *mixed Nash equilibrium if and only if the vertices of the graph G can be partitioned into two sets IS , VC (*$VC \cup IS = V$ *and* $VC \cap IS = \emptyset$*), such that IS is an independent set of G (equivalently, VC is a vertex cover of the graph) and G is a VC-expander graph.*

Proof. We first prove that if G has an independent set IS and the graph G is a VC-expander graph, where VC = V\IS, then $\Pi_E(G)$ *contains a* matching *mixed NE.* By the definition of a *VC*-expander graph, it holds that $Neigh(VC') \geq VC'$, for all $VC' \subseteq VC$. Thus, by the Marriage's Theorem 2.4, G has a matching M such that each vertex $u \in VC$ is matched into $V\backslash VC$ in M; that is there exists an edge $e = (u, v) \in M$, where $v \in V\backslash VC = IS$. Partition IS into two sets IS_1, IS_2, where set IS_1 consists of vertices $v \in IS$ for which there exists an $e = (u, v) \in M$ and $u \in VC$. Let IS_2 the remaining vertices of the set, i.e. $IS_2 = \{v \in IS : \forall\, u \in VC,\ (u, v) \notin M\}$.

Now, recall that there is no edge between any two vertices of set IS, since it is independent set, by assumption. Henceforth, since G is a connected graph, $\forall\ u \in IS_2 \subseteq IS$, there exists $e = (u, v) \in E$ and moreover $v \in V\backslash IS = VC$. Now, construct set $M_1 \subseteq E$ consisting of all those edges. That is, initially set $M := \emptyset$ and then for each $v \in IS_2$, add one edge $(u, v) \in E$ in M_1. Note that, by the construction of the set M_1, each edge of it is incident to only one vertex of IS_2. Next, construct the following configuration $\vec{\mathbf{s}}$ of $\Pi_E(G)$: Set $D_{\vec{s}}(vp) := IS$ and $D_{\vec{s}}(ep) := M \cup M_1$.

We first show that that $\vec{\mathbf{s}}$ is a *matching* configuration. Condition (1) of a matching configuration is fulfilled because $D_{\vec{s}}(vp)(= IS)$ is an independent set. We show that condition (2) of a *matching* configuration is fulfilled. Each vertex of set IS belongs either to IS_1 or to IS_2. By definition, each vertex of IS_1 is incident to only one edge of M and each vertex of IS_2 is incident to no edge in M. Moreover, by the construction of set M_1, each vertex of IS_2 is incident to exactly one edge of M_1. Thus, each vertex $v \in D_{\vec{s}}(vp)(= IS)$ is incident to only one edge of $D_{\vec{s}}(ep)(= M \cup M_1)$, i.e. condition (2) holds as well. Henceforth, $\vec{\mathbf{s}}$ is a *matching* configuration.

We next show that condition **1** of Theorem 3.2 is satisfied by $\vec{\mathbf{s}}$. We first show that $D_{\vec{s}}(ep)$ is an edge cover of G. This is true because (i) set $M \subseteq D_{\vec{s}}(ep)$ covers all vertices of set VC and IS_1, by its construction and (ii) set $M_1 \subseteq D_{\vec{s}}(ep)$ covers all vertices of set IS_2, which are the remaining vertices of G not covered by set M, also by its construction. We next show that $D_{\vec{s}}(vp)$ is a vertex cover of the subgraph of G obtained by set $D_{\vec{s}}(ep)$. By the definition of sets $IS_1,\ IS_2 \subseteq IS$, any edge $e \in M$ is covered by a vertex of set IS_1 and each edge $e \in M_1$ is covered by a vertex of set IS_2. Since $D_{\vec{s}}(ep) = M \cup M_1$, we get that all edges of the set are covered by $D_{\vec{s}}(vp) = IS_1 \cup IS_2$. This result combined with the above observation on $D_{\vec{s}}(ep)$ concludes that condition **1** of Theorem 3.2 is satisfied by $\vec{\mathbf{s}}$. Henceforth, by Claim 4.2, it can lead to a *matching* mixed NE of $\Pi_E(G)$.

We proceed to show that if G contains a matching *mixed NE, assume* $\vec{\mathbf{s}}$*, then G has an independent set IS and the graph G is a VC-expander graph, where VC = V\IS.* Define sets $IS = D_{\vec{s}}(vp)$ and $VC = V\backslash IS$. We show that these sets satisfy the above requirements for G. Note first that, set IS is an independent of

G since $D_{\vec{s}}(vp)$ is an independent set of G by condition (1) of the definition of a *matching* configuration.

We next show G contains a matching M such that each vertex of VC is matched into $V \backslash VC$ in M. Since $D_{\vec{s}}(ep)$ is an edge cover of G (condition **1** of a mixed NE of Theorem 3.2), for each $v \in VC$, there exists an edge $(u, v) \in D_{\vec{s}}(ep)$. Note that for edge (u, v), it holds that $v \in IS$, since otherwise IS would not be a vertex cover of $D_{\vec{s}}(ep)$ (Condition **1** of a mixed NE). Now, construct a set $M \subseteq E$ consisting of all those edges. That is, That is, initially set $M := \emptyset$ and then for each $v \in VC$, add one edge $(u, v) \in D_{\vec{s}}(ep)$ in M. By the construction of set M and condition (2) of a *matching* mixed NE, we get that M is a matching of G and that each vertex of VC is matched into $V \backslash VC$ in M. Thus, by the Marriage's Theorem 2.4, we get that $Neigh(VC') \geq VC'$, for all $VC' \subseteq VC$ and so G is a VC-expander and condition (2) of a matching configuration also holds in $\vec{s}$.

■

An example of a graph G with a *matching* mixed NE $\vec{s}$ is illustrated in Figure 1. Set $D_{\vec{s}}(ep)$ is denoted by bold edges and set $D_{\vec{s}}(vp)(= IS)$ (as in Theorem 4.4) by vertices with an asterisk, $*$. We remark that *not* all graphs have a *matching* mixed NE; any odd cycle is such graph; this is so because for every edge cover EC of the graph (corresponding to $D_{\vec{s}}(ep)$), there is no set $VC \subseteq V$ (corresponding to $D_{\vec{s}}(vp)$) such that VC is a vertex cover of the graph induced by EC and VC is also an independent set of G. See Figure 1(b) for an example.

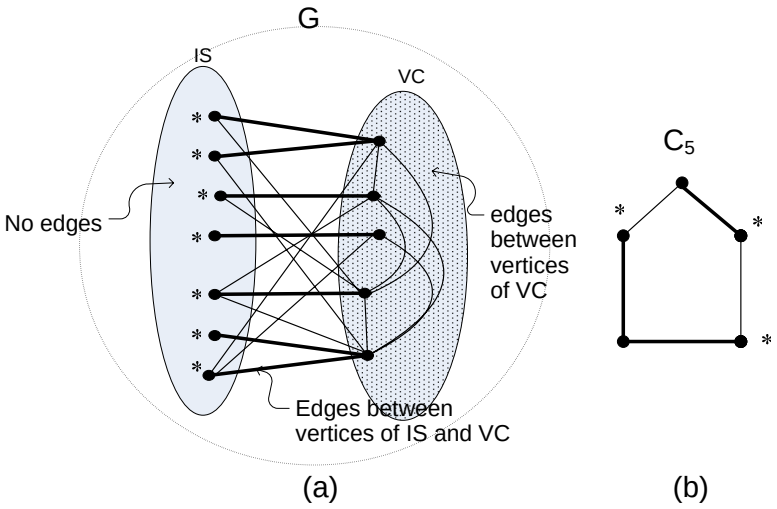

Figure 1: Examples of graphs (a) with and (b) without *matching* mixed Nash equilibrium.

4.1 A Polynomial Time Algorithm

The previous Theorems and Lemmas enabled us to develop a polynomial time algorithm for finding *matching* mixed NE for any $\Pi_E(G)$, where G is a graph satisfying the requirements of Theorem 4.4. The algorithm is described in pseudocode in Figure 2.

Theorem 4.5. *[8] Algorithm* $A(\Pi_E(G), IS, VC)$ *computes a* matching *mixed Nash equilibrium for* $\Pi_E(G)$ *in linear time* $O(n)$.

5 Bipartite Graphs

In this section we overview the basic results of [8] on the investigation the existence and polynomial time computation of *matching* mixed Nash equilibria for any $\Pi_E(G)$, for which G is a bipartite graph. We first provide some useful Lemmas and Theorems on important properties of bipartite graphs.

Lemma 5.1. *[8] In any bipartite graph G there exists a matching M and a vertex cover VC such that* **(1)** *every edge in M contains exactly one vertex of VC and* **(2)** *every vertex in VC is contained in exactly one edge of M.*

Remark 5.2. *The statement of the Lemma does not hold for all graphs; any odd cycle graph is an example of its falseness (See Figure 1(b)). The falseness of the Lemma in a general graph consists in that the statement* $(*1)$ *in its proof is false; condition (ii) required for proving* $*1$ *is not true.*

By the above Lemma 5.1, we can prove that,

Lemma 5.3. *[8] Any* X, Y*-bigraph graph G can be partitioned into two sets* IS, VC $(IS \cup VC = V$ *and* $IS \cap VC = \emptyset)$ *such that VC is a vertex cover of G (equivalently, IS is an independent set of G) and G is a VC-expander graph.*

Lemma 5.3 and Theorem 4.4 imply:

Theorem 5.4. *[8] Any* $\Pi_E(G)$ *for which G is a connected bipartite graph, contains a* matching *mixed Nash equilibrium.*

On the light of above results it is not difficult to show that,

Theorem 5.5. *[8] For any* $\Pi_E(G)$*, for which G is a bipartite graph, a* matching *mixed Nash equilibrium of* $\Pi_E(G)$ *can be computed in polynomial time,* $\max\{O(m\sqrt{n}), O(n^{2.5}/\sqrt{\log n})\}$*, using Algorithm A.*

6 Mixed Nash Equilibria in Various Graphs

Here, we overview on polynomial computable Nash equilibria of the Edge model on some practical families of graphs, such as trees, regular graphs, graphs that can be partitioned into vertex disjoint regular subgraphs, graphs with perfect matchings, showed in [9].

6.1 Trees

In Figure 3 we present in pseudocode an algorithm, called $\mathsf{Trees}(\Pi_E(T))$, for computing mixed NE for trees graph instances. The analysis following shows that the algorithm computes a *matched* NE of the instance in linear time $O(n)$. Observe that trees are bipartite graphs, thus by Theorem 5.5 a matched mixed NE of $\Pi_E(T)$ can be computed in time $O(n^{2.5}/\sqrt{\log n})$ via algorithm $A(\Pi_E(G), IS, VC)$ (section 4). Thus, algorithm $\mathsf{Trees}(\Pi_E(T))$ presented next consists a more efficient algorithm that A for computing matched NE for the case where the graph of the instance is a tree.

The proof of correctness of the Algorithm is obtained via a series of Claims proved in [9].

Claim 6.1. *Set VC, computed by Algorithm* $\mathsf{Trees}(\Pi_E(T))$*, is an independent set of* T.

Claim 6.2. *Set EC is an edge cover of* T *and VC is a vertex cover of the graph obtained by set EC.*

Claim 6.3. *Each vertex of* IS *is incident to exactly one edge of EC.*

By Claims 6.1 and 6.3 we prove,

Lemma 6.4. *Configuration* $\vec{s}^{\,t}$ *computed by algorithm* $\mathsf{Trees}(\Pi_E(T))$ *is a* matching *mixed NE.*

By the previous Lemma, combined with Claim 4.2, in the same work it is shown that,

Theorem 6.5. *For any* $\Pi_E(T)$*, where* T *is a tree graph, algorithm* $\mathsf{Trees}(\Pi_E(T))$ *computes a mixed NE in polynomial time* $O(n)$.

6.2 Regular and Polynomially Computable r-factor graphs

Theorem 6.6. *[9] For any* $\Pi_E(G)$ *for which G is an r-regular graph, a mixed NE can be computed in constant time* $O(1)$.

In the proof of the Theorem, the following configuration $\vec{s}^{\,r}$ on $\Pi_{\mathsf{E}}(G)$ is constructed:

$$\begin{array}{l} \text{For any } i \in \mathcal{N}_{vp},\ P_{\vec{s}^{\,r}}(vp_i, v) := \frac{1}{n},\ \forall v \in V(G) \text{ and then set, } \vec{s}^{\,r}_j := \vec{s}^{\,r}_i, \\ \forall j \neq i,\ j \in \mathcal{N}_{vp}\ .\ \text{Set } P_{\vec{s}^{\,r}}(ep, e) := \frac{1}{m},\ \forall e \in E. \end{array} \tag{5}$$

Then, its is shown that $\vec{s}^{\,r}$ is a mixed NE for $\Pi_{\mathsf{E}}(G)$.

The above result can be extended to graphs containing polynomially computable r-regular factors subgraphs.

Corollary 6.7. *For any $\Pi_{\mathsf{E}}(G)$ for which G is contains an r-regular factor subgraph, a mixed NE can be computed in polynomial time $O(T(G))$, where $O(T(G))$ is the time needed for the computation of G_r from G.*

Observation 6.8. *For any $\Pi_{\mathsf{E}}(G)$ for which G is a* 2-regular factor *graph, a mixed NE can be computed in polynomial time, $O(T(G))$, where $O(T(G))$ is the (polynomial) time needed for computing G_2.*

6.3 Perfect Graphs

Theorem 6.9. *[9] For any $\Pi_{\mathsf{E}}(G)$ for which G has a perfect matching, a mixed NE can be computed in linear time, $O(\sqrt{n} \cdot m)$.*

In the proof of the Theorem, first, a perfect matching M of G is computed. Then, the following configuration $\vec{s}^{\,p}$ on $\Pi_{\mathsf{E}}(G)$ is constructed:

$$\begin{array}{l} \text{For any } i \in \mathcal{N}_{vp},\ P_{\vec{s}^{\,p}}(vp_i, v) := \frac{1}{n},\ \forall v \in V(G) \text{ and set } \vec{s}^{\,p}_j := \vec{s}^{\,p}_i, \\ \forall j \neq i,\ j \in \mathcal{N}_{vp}\ .\ \text{Set } P_{\vec{s}^{\,p}}(ep, e) := \frac{1}{|M|},\ \forall e \in E. \end{array} \tag{6}$$

Then, it is shown, that both kinds of players, the vertex players and the edge player are satisfied in $\vec{s}^{\,p}$. Thus it is a mixed NE for $\Pi_{\mathsf{E}}(G)$.

7 The Price of Anarchy

In this section we overview on the basic results of [9] on the Social Cost and Price of Anarchy of the Edge model.

Lemma 7.1. *For any $\Pi_{\mathsf{E}}(G)$ and an associated mixed NE $\vec{s}^{\,*}$, the social cost $\mathsf{SC}(\Pi_{\mathsf{E}}(G), \vec{s}^{\,*})$ is upper and lower bounded as follows:*

$$\max\left\{\frac{\nu}{|D_{\vec{s}^{\,*}}(ep)|}, \frac{\nu}{|V(D_{\vec{s}^{\,*}}(vp))|}\right\} \ \leq\ \mathsf{SC}(\Pi_{\mathsf{E}}(G), \vec{s}^{\,*}) \leq \frac{\Delta(D_{\vec{s}^{\,*}}(ep)) \cdot \nu}{|D_{\vec{s}^{\,*}}(ep)|} \tag{7}$$

These bounds are tight.

Theorem 7.2. *The Price of Anarchy for the Edge model is $\frac{n}{2} \leq \mathsf{r}(\mathsf{E}) \leq n$.*

8 The Path Model

In the last section, we take a glimpse on the Path model. In [9], we provide the following characterization of pure Nash Equilibria in the Path model.

Theorem 8.1. *For any graph G, $\Pi_P(G)$ has a pure NE if and only if G contains a hamiltonian path.*

This characterization implies the following result regarding the existence of pure NE.

Corollary 8.2. *The problem of deciding whether there exists a pure NE for any $\Pi_P(G)$ is $\mathcal{NP}$-complete.*

References

[1] N. Alon, R. M. Karp, D. Peleg and D. West, "A Graph-Theoretic Game and its Application to the *k*-Server Problem", *SIAM Journal on Computing*, Vol 24, No 1, pp. 78-100, February 1995.

[2] J. Aspnes, K. Chang and A. Yampolskiy, "Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Problem", *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 43-52, January 2005.

[3] A. S. Asratian, D. Tristan and M. J. Häggkvist, *Bipartite Graphs and Their Applications*, Cambridge Tracts in Mathematics, 131, 1998.

[4] M. Franklin, P. Alto, Z. Galil and Moti Yung, "Eavesdropping Games: a Graph-Theoretic Approach to Privacy in Distributed Systems", *Journal of the ACM*, Vol 47, No 2, pp. 225-243, March 2000.

[5] M. Kearns and L. Ortiz, "Algorithms for Interdependent Security Games", *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*, December 2003.

[6] E. Koutsoupias and C. H. Papadimitriou, "Worst-Case Equilibria", *In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404–413, Springer-Verlag, March 1999.

[7] B. Manthey, "On Approximating Restricted Cycle Covers", Technical Report *arXiv:cs.CC/0504038 v2* , June 10, 2005.

[8] M. Mavronicolas, V. Papadopoulou, A. Philippou, P. Spirakis, "A Network Game with Attacker and Protector Entities", *In the Proceedings of the 16th Annual International Symposium on Algorithms and Computation*, 2005.

[9] M. Mavronicolas, V. Papadopoulou, A. Philippou, P. Spirakis, "A Graph-Theoretic Network Security Game", *In the Proceedings of the 1st Workshop on Internet and Network Economics*, 2005.

[10] S. Micali and V.V. Vazirani, "An $O(V^{1/2}E)$ Algorithm for Finding Maximum Matching in General Graphs", *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science*, pp. 17-27, 1980.

[11] J. F. Nash, "Equilibrium Points in n-Person Games", *Proceedings of the National Acanemy of Sciences of the United States of America*, Vol 36, pp 48-49, 1950.

[12] J. F. Nash, "Non cooperative Games", *Annals of Mathematics*, Vol 54, No 2, pp. 286-295, 1951.

[13] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, MIT Press, 1994.

[14] C. H. Papadimitriou, "Algorithms, Games, and the Internet", *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pp. 749-753, June 2001.

[15] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Third Edition, Prentice Hall, 2003.

[16] W. T. Tutte, "A Short Proof of the Factor Theorem for Finite Graphs", *Canadian Journal of Mathematics*, Vol 6, pp. 347-352, 1954.

Algorithm $A(\Pi_E(G), IS, VC)$

INPUT: A game $\Pi_E(G)$ and a partition of $V(G)$ into sets IS, $VC = V \backslash IS$, such that IS is an independent set of G and G is a VC-expander graph.
OUTPUT: A mixed NE $\vec{s}$ for $\Pi_E(G)$.

1. Compute a set $M \subseteq E$ as follows:
   (a) *Initialization*: Set $M := \emptyset$, $Matched := \emptyset$ (currently matched vertices in M), $Unmatched := VC$ (currently unmatched vertices of VC in M), $Unused := IS$, $i := 1$, $G_i := G$ and $M_1 := \emptyset$.
   (b) While $Unmatched \neq \emptyset$ Do:
      i. Consider a $u \in Unmatched$.
      ii. Find a $v \in Unused$ such that $(u, v) \in E_i$. Set $M := M \cup (u, v)$, $Unused := Unused \backslash \{v\}$.
      iii. *Prepare next iteration*: Set $i := i + 1$, $Matched := Matched \cup \{u\}$, $Unmatched := Unmatched \backslash \{u\}$, $G_i := G_{i-1} \backslash u \backslash v$.
2. Partition set IS into two sets IS_1, IS_2 as follows: $IS_1 := \{u \in IS : \exists\ (u, v) \in M\}$ and $IS_2 := IS \backslash IS_1$. Note that $IS_2 := \{u \in IS : \forall v \in VC,\ \nexists\ (u, v) \in M\}$.
   Compute set M_1 as follows: $\forall\ u \in IS_2$, set $M_1 := M_1 \cup (u, v)$, for any $(u, v) \in E$, $v \in VC$.
3. Define a configuration $\vec{s}$ with the following support: $D_{\vec{s}}(vp) := IS$, $D_{\vec{s}}(ep) := M \cup M_1$.
4. Determine the probabilities distributions of the vertex players and the e.p. of configuration $\vec{s}$ using equations (3) and (4) of Claim 4.2.

Figure 2: Algorithm $A(\Pi_E(G), IS, VC)$.

Algorithm Trees($\Pi_E(T)$)

1. Initialization: $VC := \emptyset$, $EC := \emptyset$, $r := 1$, $T_r := T$.

2. Repeat until $T_r == \emptyset$

   (a) Find the leaves of the tree T_r, $leaves(T_r)$.

   (b) Set $VC := VC \cup leaves(T_r)$.

   (c) For each $v \in leaves(T_r)$ do:

   If $parent_{T_r}(v) \neq \emptyset$, then $EC := EC \cup \{(v, parent_{T_r}(v))\}$,
   else $EC := EC \cup \{(v, u)\}$, for any $u \in children_T(v)$.

   (d) Update tree: $T_{r+1} := T_r \backslash leaves(T_r) \backslash parents(leaves(T_r))$.
   Set $r := r + 1$.

3. Define a configuration $\vec{s}^{\,t}$ with the following support:
   For any $i \in \mathcal{N}_{vp}$, set $D_{\vec{s}^{\,t}}(vp_i) := VC$ and $D_{\vec{s}^{\,t}}(ep) := EC$. Then set $D_{\vec{s}^{\,t}}(vp_j) := D_{\vec{s}^{\,t}}(vp_i)$, $\forall\ j \neq i,\ j \in \mathcal{N}_{vp}$.

4. Determine the probabilities distributions of players in $\vec{s}^{\ t}$ as follows:
   ep : $\forall\ e \in D_{\vec{s}^{\,t}}(ep)$, set $P_{\vec{s}^{\,t}}(ep, e) := 1/|EC|$. Also, $\forall\ e' \in E(T)$, $e' \notin D_{\vec{s}^{\,t}}(ep)$, set $P_{\vec{s}^{\,t}}(ep, e') := 0$.

   For any vp_i, $i \in \mathcal{N}_{vp}$: $\forall\ v \in D_{\vec{s}^{\,t}}(vp_i)$, set $P_{\vec{s}^{\,t}}(vp_i, v) := \frac{1}{|VC|}$. Also, $\forall\ u \notin D_{\vec{s}^{\,t}}(vp_i)$, set $P_{\vec{s}^{\,t}}(vp_i, u) := 0$. Then set $\vec{s}^{\ t}_j = \vec{s}^{\ t}_i$, $\forall\ j \neq i,\ j \in \mathcal{N}_{vp}$.

Figure 3: Algorithm Trees($\Pi_E(T)$).