Computational Game Theory: An Introduction

Panagiota N. Panagopoulou and Paul. G. Spirakis

2007
1 Introduction

Game Theory was founded by von Neumann and Morgenstern and can be defined as the study of mathematical models of interactive decision making. Game Theory provides general mathematical techniques for analyzing situations in which two or more individuals, called players, make decisions that will influence one another’s welfare. The dominant aspect of Game Theory is the belief that each player is
rational, in the sense that she makes decisions consistently in pursuit of her own objectives, and this rationality is common knowledge.

This chapter is an introduction to the basic concepts and advances of a new field, that of Computational (or Algorithmic) Game Theory. This field is a fertile interaction between the very deep and older field of Game Theory on the one hand, and of the younger field of Algorithms and Complexity on the other.

In the seminal paper of [Papadimitriou, 2001], the interaction between Game Theory and Theoretical Computer Science was considered as a potential “object of the next great modeling adventure” of the field of computer scientists. In that work, it was pointed out that a fusion of algorithmic ideas and game theoretic concepts might yield to the most appropriate mathematical tools and insights for understanding the socio-economic complexity of the Internet. Furthermore, some game theoretic concepts (e.g. Nash equilibria) are related to fundamental computational issues, such as finding solutions that are guaranteed to exist via non-constructive methods, or approximating them. On the other hand, game theoretic concepts have been used by theoretical computer scientists as a means of studying the computational complexity of algorithms: proving lower bounds can be seen as a game between the algorithm designer and an adversary.

In this chapter we deal with some basic topics of Computational Game Theory. Namely, we study the computational complexity of Nash equilibria (which constitute the most important solution concept in Game Theory) and review the related algorithms proposed in the literature. Then, given the apparent difficulty of computing exact Nash equilibria, we study the efficient computation of approximate notions of Nash equilibria. Next we deal with several computational issues related to the class of congestion games, which model the selfish behavior of individuals.
when competing on the usage of a common set of resources. Finally, we study the price of anarchy (in the context of congestion games), which is defined as a measure of the performance degradation due to the lack of coordination among the involved players.

As this new field of Computational Game Theory evolves, important research results have already appeared in several directions that are not touched here, because of lack of space and because of our selection of what are the most basic topics. For example, Mechanism Design, Cooperative Games, Bayesian Games and their computational issues are not considered in this chapter. We hope that the interested reader will be motivated by this chapter to also look into those new research lines.

2 Underlying Principles

2.1 Strategic Form Games

In the language of Game Theory, a game refers to any situation in which two or more decision-makers interact. In this chapter we focus on finite games in strategic form:

**Definition 2.1** A finite strategic form game is any \( \Gamma \) of the form

\[
\Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle ,
\]

where \( N \) is a finite nonempty set, and, for each \( i \in N \), \( C_i \) is a finite nonempty set and \( u_i \) is a function from \( C = \times_{j \in N} C_j \) into the set of real numbers \( \mathbb{R} \).

In the above definition, \( N \) is the set of players in the game \( \Gamma \). For each player \( i \in N \), \( C_i \) is the set of actions available to player \( i \). When the strategic form game
Γ is played, each player \( i \) must choose one of the actions in the set \( C_i \). For each combination of actions \( c = (c_j)_{j \in N} \in C \) (one for each player), the number \( u_i(c) \) represents the payoff that player \( i \) would get in this game if \( c \) were the combination of actions implemented by the players. When we study a strategic form game, we assume that all the players choose their actions simultaneously.

Given any strategic form game \( \Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle \), a (mixed) strategy \( \sigma_i \) for any player \( i \in N \) is a probability distribution over her set of actions \( C_i \). Therefore, for each \( c_i \in C_i \), the number \( \sigma_i(c_i) \) represents the probability that player \( i \) would choose \( c_i \). A pure strategy for player \( i \) is a strategy that poses probability 1 to a specific action in \( C_i \) and zero to the rest; we slightly abuse notation and denote \( c_i \) the pure strategy that poses probability 1 to action \( c_i \in C_i \). Let \( \Delta(C_i) \) denote the set of all possible strategies of player \( i \), i.e. the set of all possible distributions on \( C_i \).

The support of strategy \( \sigma_i \in \Delta(C_i) \) is the subset of actions of player \( i \) that are assigned strictly positive probability:

\[
\text{support}(\sigma_i) = \{c_i \in C_i \mid \sigma_i(c_i) > 0\} .
\]  

(2)

A strategy profile \( \sigma = (\sigma_i)_{i \in N} \) is a combination of strategies, one for each player, so the set of all strategy profiles is \( \times_{j \in N} \Delta(C_j) \).

For any strategy profile \( \sigma \), let \( u_i(\sigma) \) denote the expected payoff that player \( i \) would get when the players independently choose their actions according to \( \sigma \). That is,

\[
u_i(\sigma) = \sum_{c \in C} \left( \prod_{j \in N} \sigma_j(c_j) \right) u_i(c) \quad \forall i \in N .
\]  

(3)

For any strategy profile \( \sigma \), any player \( i \in N \), and any \( \tau_i \in \Delta(C_i) \), we denote \((\tau_i, \sigma_{-i})\) the strategy profile in which the \( i \)-th component is \( \tau_i \) while all the other
components are as in $\sigma$. Thus

$$u_i(\tau_i, \sigma_{-i}) = \sum_{c \in C} \left( \prod_{j \in N \setminus \{i\}} \sigma_j(c_j) \right) \tau_i(c_i) u_i(c) \quad \forall i \in N . \quad (4)$$

A game $\Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle$ is symmetric if all players share the same action set, and the payoffs for playing a particular strategy depend only on the other strategies employed, not on who is playing them. Formally:

**Definition 2.2** A finite game in strategic form $\Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle$ is symmetric if $C_i = C_j = \hat{C}$ for all $i, j \in N$, and $u_i(c_i, c_{-i}) = u_j(c_j, c_{-j})$ for $c_i = c_j \in \hat{C}$ and $c_{-i} = c_{-j} \in \hat{C} | N | - 1$, for all $i, j \in N$.

### 2.2 Nash Equilibria

The most important solution concept in a strategic game is the **Nash equilibrium** [Nash, 1951]. A Nash equilibrium is a strategy profile $\hat{\sigma}$ that corresponds to a steady state, in the sense that no player has a reason to change her strategy if everyone else adheres to $\hat{\sigma}$.

**Definition 2.3** Given a strategic form game $\Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle$, a strategy profile $\hat{\sigma}$ is a Nash equilibrium of $\Gamma$ if

$$u_i(\hat{\sigma}) \geq u_i(\sigma_i, \hat{\sigma}_{-i}) \quad \forall i \in N , \quad \forall \sigma_i \in \Delta(C_i) , \quad (5)$$

or equivalently

$$u_i(\hat{\sigma}) \geq u_i(c_i, \hat{\sigma}_{-i}) \quad \forall i \in N , \quad \forall c_i \in C_i , \quad (6)$$

or equivalently

$$\hat{\sigma}_i(c_i) > 0 \implies c_i \in \arg \max_{d_i \in C_i} u_i(d_i, \hat{\sigma}_{-i}) \quad \forall i \in N , \quad \forall c_i \in C_i . \quad (7)$$
Thus, in a Nash equilibrium \( \hat{\sigma}, \hat{\sigma}_i \) is a best reply to \( \hat{\sigma}_{-i} \) for all \( i \in N \), in the sense that it maximizes the expected payoff received by player \( i \) given the strategies \( \hat{\sigma}_{-i} \) chosen by the rest of the players.

We can now state the general existence theorem of [Nash, 1951]:

**Theorem 2.1** Given any finite game in strategic form \( \Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle \), there exists at least one Nash equilibrium \( \hat{\sigma} \in \times_{i \in N} \Delta(C_i) \).

A Nash equilibrium is pure if all players’ strategies are pure.

### 2.3 Bimatrix games

**Bimatrix games** are 2-player games such that the payoff functions can be described by two real matrices \( R \) and \( C \). In particular, an \( n \times m \) bimatrix game \( \Gamma = \langle R, C \rangle \) is a 2-player strategic game, where \( R \) and \( C \) are \( n \times m \) matrices. The \( n \) rows of \( R \) and \( C \) correspond to the actions of player 1 (the row player) and the \( m \) columns of \( R \) and \( C \) correspond to the actions of player 2 (the column player).

Denote \( [n] \) the set of actions available to the row player, i.e. \( [n] = \{1, 2, \ldots, n\} \). Similarly, let \( [m] = \{1, 2, \ldots, m\} \) be the action set of the column player. For any matrix \( A \), denote \( a_{ij} \) the element in the \( i \)th row and \( j \)th column of \( A \). Then the payoff that the players receive in the pure strategy profile \( (i, j) \in [n] \times [m] \) are \( r_{ij} \) for the row player and \( c_{ij} \) for the column player.

A strategy \( x = (x_1 \ldots x_n)^T \) for the row player is a probability distribution on rows, written as an \( n \times 1 \) vector of probabilities. We denote \( \mathbb{P}^n \) the set of all probability vectors in \( \mathbb{R}^n \), i.e.

\[
\mathbb{P}^n = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1 \text{ and } x_i \geq 0 \text{ } \forall i \in [n] \right\}.
\] (8)
Similarly, a strategy for the column player is any \( y \in P^m \). The expected payoff that the players receive in the strategy profile \((x, y)\) are \( x^T Ay \) for the row player and \( x^T By \) for the column player.

In the following, we denote by \( e_i \) the probability vector in which the \( i \)th component is 1 and all other components are zero. The size of \( e_i \) will be clear from the context.

Using the above notation, we restate the definition of a Nash equilibrium for the case of a bimatrix game:

**Definition 2.4** A strategy profile \((x, y)\) is a Nash equilibrium for the \( n \times m \) bimatrix game \( \Gamma = \langle R, C \rangle \) if

1. For all \( i \in [n] \), \( e_i^T R y \leq x^T R y \) and
2. For all \( j \in [m] \), \( x^T C e_j \leq x^T C y \).

While in a Nash equilibrium no player can increase her payoff by unilaterally changing her strategy, in an \( \epsilon \)-Nash equilibrium no player can increase her payoff by more than \( \epsilon \) by unilaterally changing her strategy:

**Definition 2.5** For any \( \epsilon \geq 0 \) a strategy profile \((x, y)\) is an \( \epsilon \)-Nash equilibrium for the \( n \times m \) bimatrix game \( \Gamma = \langle R, C \rangle \) if

1. For all \( i \in [n] \), \( e_i^T R y \leq x^T R y + \epsilon \) and
2. For all \( j \in [m] \), \( x^T C e_j \leq x^T C y + \epsilon \).

A stronger notion of approximate Nash equilibrium is the \( \epsilon \)-well supported Nash equilibrium (or well-supported \( \epsilon \)-approximate Nash equilibrium). This is an \( \epsilon \)-Nash equilibrium with the additional property that each player plays only approximately best-response pure strategies with nonzero probability:
Definition 2.6 (\(\epsilon\)-well-supported Nash equilibrium) For any \(\epsilon \geq 0\) a strategy profile \((x, y)\) is an \(\epsilon\)-well-supported Nash equilibrium for the \(n \times m\) bimatrix game \(\Gamma = (R, C)\) if

1. For all \(i \in [n]\),
\[
x_i > 0 \implies e_i^T R y \geq e_k^T R y - \epsilon \quad \forall k \in [n]
\]
2. For \(j \in [m]\),
\[
y_j > 0 \implies x^T C e_j \geq x^T C e_k - \epsilon \quad \forall k \in [m]
\]

Note that both notions of approximate equilibria are defined with respect to an additive error term \(\epsilon\). Although (exact) Nash equilibria are known not to be affected by any positive scaling of the payoff matrices, it is important to mention that approximate notions of Nash equilibria are indeed affected. Therefore, the commonly used assumption in the literature when referring to approximate Nash equilibria is that the bimatrix game is positively normalized, i.e. both matrices entries lie in \([0, 1]\), and this assumption is adopted here.

3 Computational Complexity of Nash Equilibria

3.1 The Complexity Class PPAD

The complexity class PPAD (standing for Polynomial Parity Argument in a Directed graph), introduced in [Papadimitriou, 1994], is described in terms of an algorithm which implicitly defines a finite, though exponentially large, directed graph consisting of lines and cycles and having a standard source node. We are asking for any node, other than the standard source, with indegree + outdegree =
1. The existence of such a node is established via the graph-theoretic parity argument in a directed graph, which states that a directed graph $G$, each node of which has indegree at most 1 and outdegree at most 1, is a disjoint union of paths and cycles. Specifically, the number of endpoints (i.e. sources and sinks) must be even. Thus, given one endpoint, there must be another. A problem in PPAD is essentially a search problem for which the solution sought is guaranteed to exist by an inefficiently constructive proof.

PPAD is defined by its complete problem Another End Of Directed Lines (or AEL in short) [Papadimitriou, 1994]: The input instance of the problem is a directed graph $G$ of an exponential number of vertices, each with at most one incoming edge and at most one outgoing edge. A starting vertex $0$ with indegree 0 and outdegree 1 is given, and the output is a sink node, or a source other than $0$. However the graph is not directly given as the input, since then the problem would be solvable in polynomial time. Instead, the graph is generated by a given boolean circuit of polynomial size (e.g. $\log |G|$).

Formally, AEL is defined as follows. The input instance is a pair $(C, 0^n)$ where $C$ is a boolean circuit with $n$ input bits. For each $x \in \{0, 1\}^n$, let $C(x)$ denote the output of $C$ when the input is $x$. A directed graph $G = (V, E)$ with $V = \{0, 1\}^n$ can be generated by $C$ as follows:

1. For each vertex $v \in V$, $C(v)$ is an ordered pair $(u, w)$. Each of $u$ and $w$ is either a vertex in $V$ or nil.

2. Edge $(u, v) \in E$ if and only if $v$ is the second component of $C(u)$ and $u$ is the first component of $C(v)$.

$C$ must satisfy $C(0^n) = (nil, u \neq nil)$ and the first component of $C(u)$ is $0^n$. 


These conditions on $C$ imply that $0^n$ is a source of $G$, and we need to find either a source or a sink in $G$ other than $0^n$.

A search problem is in **PPAD** if it can be reduced in polynomial time to $AEL$. Moreover, it is **PPAD**-complete if there is a polynomial time reduction from $AEL$ to it.

### 3.2 Complexity of Nash Equilibrium

Despite the certain existence of Nash equilibria in finite games, the computational complexity of finding a Nash equilibrium used to be a wide open problem for several years. Call $r$-NASH the problem of computing a Nash equilibrium in a game with $r$ players. [Daskalakis et al., 2006a] showed that $r$-NASH is **PPAD**-complete for all $r \geq 4$. The proof used a reduction from a new three-dimensional discrete fixed point problem named 3-DIMENSIONAL BROUWER (that was proved to be **PPAD**-complete) to a particular type of game. Two concurrent and independent works, [Chen and Deng, 2005] and [Daskalakis and Papadimitriou, 2005], showed that 3-Nash is **PPAD**-complete as well; the former used a direct reduction from 4-Nash while the latter used a variant of the proof in [Daskalakis et al., 2006a].

Finally, in [Chen and Deng, 2006] it was shown that 2-Nash is **PPAD**-complete via a reduction from 3-DIMENSIONAL BROUWER, which is defined as follows:

**3-DIMENSIONAL BROUWER**: The input is a pair $(C, 0^n)$ where $C$ is a circuit with $3n$ input bits and 6 output bits $\Delta^+_1, \Delta^-_1, \Delta^+_2, \Delta^-_2, \Delta^+_3, \Delta^-_3$. It specifies a function $\phi$ of a very special form. Let

$$A^n = \{ r \in \mathbb{Z}^3 \mid 0 \leq r_i \leq 2^n - 1, i = 1, 2, 3 \}$$  \hspace{1cm} (9)
and

\[ B^n = \{ r \in A^n \mid \exists i : r_i = 0 \text{ or } r_i = 2^n - 1 \} \]  \hspace{1cm} (10)

For each \( r \in A^n \), we define a cubelet in \([0, 1]^3\) as

\[ \{ q \in \mathbb{R}^3 \mid r_i 2^{-n} \leq q_i \leq (r_i + 1)2^{-n}, i = 1, 2, 3 \} \]  \hspace{1cm} (11)

and let \( c_r \) denote its center point. Function \( \phi \) is defined on the set of \( 23^n \) centers. For every center \( c_r \), where \( r \in A^n \), \( \phi(c_r) \in \{ e^1, e^2, e^3, e^4 \} \subset \mathbb{Z}^3 \) and is specified by the output bits of \( C(r) \) as follows:

- \( \Delta^+_1 = 1 \), other five bits are 0: \( \phi(c_r) = e^1 = (1, 0, 0) \)
- \( \Delta^+_2 = 1 \), other five bits are 0: \( \phi(c_r) = e^2 = (0, 1, 0) \)
- \( \Delta^+_3 = 1 \), other five bits are 0: \( \phi(c_r) = e^3 = (0, 0, 1) \)
- \( \Delta^-_1 = \Delta^-_2 = \Delta^-_3 = 1 \), other three bits are 0: \( \phi(c_r) = e^4 = (-1, -1, -1) \).

For all \( r \in A^n \), the six output bits of \( C(r) \) are guaranteed to fall into one of the four cases above. \( C \) also satisfies the following boundary condition: For each \( r \in B^n \), if there exists \( 1 \leq i \leq 3 \) such that \( r_i = 0 \), letting \( \ell \) be the largest index such that \( r_\ell = 0 \), then \( \phi(c_r) = e^\ell \); otherwise, \( \phi(c_r) = e^4 \). A vertex of the cubelet is called panchromatic if, among the eight cubelets adjacent to it, there are four that have all four vectors \( e^1, e^2, e^3 \) and \( e^4 \). The output of 3-DIMENSIONAL BROUWER is a panchromatic vertex of \( \phi \).

Now let \( U = (C, 0^n) \) be an input instance of 3-DIMENSIONAL BROUWER and let \( m \) be the smallest integer such that \( K = 2^m > (|C| + n)^2 \). [Chen and Deng, 2006] show how to construct a two-player game \( G^U \) in which both players have \( 2K \) strategies and prove that every \( \epsilon \)-Nash equilibrium \((x, y)\) of \( G^U \) must satisfy a set
of constraints. These constraints imply that, given any \( \epsilon \)-Nash equilibrium of game \( G^U \) where \( \epsilon = 2^{-(m+4n)} \), a panchromatic vertex of function \( \phi \) can be computed in polynomial time. This implies that 3-DIMENSIONAL BROWWER reduces to 2-NASH.

4 Algorithms for Computing Nash Equilibria

4.1 \( n \)-Person Games

We will now describe a general procedure for finding Nash equilibria of any finite strategic form game \( \Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle \). Although there are infinitely many strategy profiles, there is only a finite number of subsets of \( C \) that can be supports of Nash equilibria, so we can find all Nash equilibria of \( \Gamma \) by exhaustively searching over all possible supports.

For each player \( i \in N \), let \( D_i \) be some nonempty subset of \( C_i \), representing the set of actions of \( i \) that are assigned positive probability. If there exists a Nash equilibrium \( \sigma \) with support \( \times_{i \in N} D_i \), then there must exist numbers \( \omega_i \in \mathbb{R} \) for all \( i \in N \) such that the following equations are satisfied:

\[
\sum_{c_{-i} \in C_{-i}} \left( \prod_{j \in N \setminus \{i\}} \sigma_j(c_j) \right) u_i(d_i, c_{-i}) = \omega_i \quad \forall i \in N, \quad \forall d_i \in D_i \quad (12)
\]

\[
\sigma_i(e_i) = 0 \quad \forall i \in N, \quad \forall e_i \in C_i \setminus D_i \quad (13)
\]

\[
\sum_{e_i \in D_i} \sigma_i(c_i) = 1 \quad \forall i \in N . \quad (14)
\]

Condition (12) asserts that each player gets the same payoff \( \omega_i \) if she chooses any of her actions that are assigned positive probability. Conditions (13) and (14) follow from the assumption that \( \sigma \) is a strategy profile with support \( \times_{i \in N} D_i \).
These three conditions give us \( \sum_{i \in N} (|C_i| + 1) \) equations in the same number of unknowns (namely, the payoffs \( \omega_i \) and the probabilities \( \sigma_i(c_i) \) for each \( i \in N, c_i \in C_i \)). For games with more than two players, Equation (12) becomes nonlinear in \( \sigma \); however we can still solve the system of equations (12)–(14).

Note however that the solutions of (12)–(14) do not necessarily correspond to Nash equilibria of \( \Gamma \) since there are three difficulties that may arise. First, no solution may exist. Second, a solution might not correspond to a strategy profile \( \sigma \) if \( \sigma_i(d_i) < 0 \) for some \( i \in N, d_i \in D_i \). So we must require

\[
\sigma_i(d_i) \geq 0 \quad \forall i \in N, \forall d_i \in D_i . \tag{15}
\]

Third, a solution may fail to be an equilibrium if there exists a strategy \( e_i \in C_i \setminus D_i \) for some player \( i \in N \) that would give player \( i \) a better payoff against \( \sigma_{-i} \). So we must require

\[
\omega_i \geq u_i(e_i, \sigma_{-i}) \quad \forall i \in N , \quad \forall e_i \in C_i \setminus D_i . \tag{16}
\]

Any solution \( ((\sigma_i)_{i \in N}, (\omega_i)_{i \in N}) \) to equations (12)–(14) that also satisfies (15) and (16) is a Nash equilibrium of \( \Gamma \). Furthermore, if there is no solution that satisfies (12)–(16) then there is no Nash equilibrium with support \( \times_{i \in N} D_i \). Nash’s existence theorem guarantees that, if we exhaustively search over all possible \( \times_{i \in N} |C_i| \) supports, we will find at least one support \( \times_{i \in N} D_i \) for which (12)–(16) are satisfied.

4.2 Two Person Games

Consider an \( n \times m \) bimatrix game \( \Gamma = \langle R, C \rangle \). Let \( N = [n] \) and \( M = [m] \) denote the action set of the row and column player respectively. Given a fixed
strategy $y \in \mathbb{P}^m$ for the column player, a best-response of the row player to $y$ is a probability vector $x \in \mathbb{P}^n$ that maximizes the expression $x^T R y$. Therefore, $x$ is a solution to the Linear Program

$$\begin{align*}
\text{maximize} & \quad x^T R y \\
\text{subject to} & \quad \sum_{i \in N} x_i = 1 \\
& \quad x_i \geq 0 \quad \forall i \in N .
\end{align*} \tag{17}$$

The dual of the Linear Program (17) is

$$\begin{align*}
\text{minimize} & \quad u \\
\text{subject to} & \quad u \geq (R y)_i \quad \forall i \in N .
\end{align*} \tag{18}$$

By the strong duality theorem of linear programming, (17) and (18) have the same optimal value. Let

$$E = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix} \in \mathbb{R}^{1 \times N} \tag{19}$$

$$e = 1 . \tag{20}$$

Then, a feasible solution $x$ is optimal if and only if there is a dual solution $u$ fulfilling $u \geq (R y)_j$ for all $j \in M$ and $x^T R y = u$, that is $x^T R y = x^T E^T u$, or equivalently

$$x^T (E^T u - R y) = 0 . \tag{21}$$

Since $x$ and $E^T u - R y$ are non negative, (21) states that the have to be complementary in the sense that they can not both have positive components in the same position. Therefore, for each positive component of $x$, the respective component of $E^T u - R y$ is zero and $u$ is the maximum of the components in $R y$. Thus any
pure strategy \( i \in N \) of the row player is a best response to \( y \) if and only if the \( i \)th component of \( E^T u - Ry \) is zero.

Similarly, given a fixed strategy \( x \in P^n \) for the row player, a best-response of the column player to \( x \) is a probability vector \( y \in P^n \) that maximizes the expression \( x^T C y \). Therefore, \( y \) is a solution to the Linear Program

\[
\begin{align*}
\text{maximize} & \quad x^T C y \\
\text{subject to} & \quad \sum_{j \in M} y_j = 1 \\
& \quad y_j \geq 0 \quad \forall j \in M .
\end{align*}
\] (22)

The dual of the Linear Program (22) is

\[
\begin{align*}
\text{minimize} & \quad v \\
\text{subject to} & \quad v \geq (C^T x)_j \quad \forall j \in M .
\end{align*}
\] (23)

Now let

\[
\begin{align*}
F &= \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix} \in \mathbb{R}^{1 \times M} \\
f &= 1 .
\end{align*}
\] (24)

(25)

Here, a primal-dual pair \((y, v)\) of feasible solutions is optimal if and only if

\[
y^T (F^T v - C^T x) = 0 .
\] (26)

The above conditions for both players yield:

**Theorem 4.1** The bimatrix game \( \Gamma = \langle R, C \rangle \) has the Nash equilibrium \((x, y)\) if
and only if for suitable \( u, v \in \mathbb{R} \)

\[
Ex = e \quad \text{(27)}
\]
\[
Fy = f \quad \text{(28)}
\]
\[
(E^Tu - Ry)_i \geq 0 \quad \forall i \in N \quad \text{(29)}
\]
\[
(F^Tv - C^Tx)_j \geq 0 \quad \forall j \in M \quad \text{(30)}
\]
\[
x_i \geq 0 \quad \forall i \in N \quad \text{(31)}
\]
\[
y_j \geq 0 \quad \forall j \in M \quad \text{(32)}
\]

and (21), (26) hold.

The conditions in the above theorem define a special case of a linear complementarity problem (LCP) [von Stengel, 2002]. The most important method for finding a solution of the LCP defined by the theorem is the Lemke-Howson algorithm, which we will describe later in this section.

4.2.1 Zero-Sum Games

Consider the case where \( \Gamma = \langle R, C \rangle \) is a zero-sum game, that is \( C = -R \). If the row player chooses strategy \( x \), she can be sure of winning only \( \min_y x^TRy \). Thus, the optimal choice for the row player is given by \( \max_x \min_y x^TRy \). Similarly, the optimal choice for the column player is given by \( \max_y \min_x x^TCy = \min_y \max_x x^TRy \).

Then, the problem of computing the column player’s optimal strategy can be
expressed as

\[
\begin{align*}
\text{minimize} \quad & u \\
\text{subject to} \quad & u - (Ry)_i \geq 0 \quad \forall i \in N \\
& \sum_{j \in M} y_j = 1 \\
& y_j \geq 0 \quad \forall j \in M.
\end{align*}
\] (33)  

The dual of this Linear Program has the form

\[
\begin{align*}
\text{maximize} \quad & v \\
\text{subject to} \quad & v - (RTx)_j \leq 0 \quad \forall j \in M \\
& \sum_{i \in N} x_i = 1 \\
& x_i \geq 0 \quad \forall i \in N.
\end{align*}
\] (34)  

It is easy to verify that the above Linear Program describes the problem of finding an optimal strategy for the row player. Therefore, the problem of computing a Nash equilibrium for a zero-sum game is solvable in polynomial time. Moreover, by strong LP duality, \( \max_x \min_y x^T R y = \min_y \max_x x^T R y. \)

### 4.2.2 The Lemke-Howson algorithm

In this section we present the classical algorithm by [Lemke and Howson, 1964] that computes a Nash equilibrium of a nondegenerate bimatrix game \( \Gamma = (R, C). \)

**Definition 4.1** A bimatrix game \( \Gamma = (R, C) \) is called nondegenerate if the number of pure best responses to a strategy never exceeds the size of its support.

It is useful to assume that the actions sets of the players are disjoint. In particular, we assume that the action set of the row player is \( N = \{1, \ldots, n\} \) and the
action set of the column player is \( M = \{n + 1, \ldots, n + m\} \). For each \( i \in N \) and \( j \in M \), define

\[
X(i) = \{x \in \mathbb{P}^n | x_i = 0\} \quad (37)
\]

\[
X(j) = \left\{ x \in \mathbb{P}^n \mid \sum_{i=1}^{n} c_{ij} x_i \geq \sum_{i=1}^{n} c_{ik} x_i \quad \forall k \in M \right\} \quad (38)
\]

\[
Y(j) = \{y \in \mathbb{P}^m | y_j = 0\} \quad (39)
\]

\[
Y(i) = \left\{ y \in \mathbb{P}^m \mid \sum_{j=1}^{m} r_{ij} y_j \geq \sum_{j=1}^{m} r_{kj} y_j \quad \forall k \in N \right\} . \quad (40)
\]

Any strategy \( x \in \mathbb{P}^n \) and \( y \in \mathbb{P}^m \) is labelled with certain elements of \( N \cup M \) as follows: for each \( k \in N \cup M \), \( x \) has label \( k \) if \( x \in X(k) \) and \( y \) has label \( k \) if \( y \in Y(k) \). So the labels of a strategy \( x \) (\( y \)) of the row (column) player denote the actions of the row (column) player that are assigned zero probability and the actions of the column (row) player that are best responses to \( x \) (\( y \)). Since, in any Nash equilibrium of the game, any action of a player is either a best response to the opponent’s strategy or is played with probability zero, it follows that a strategy profile \( (x, y) \) is a Nash equilibrium if and only if \( x \) and \( y \) are completely labelled, i.e. the union of the labels of \( x \) and \( y \) is the set \( N \cup M \).

**Theorem 4.2** A strategy profile \( (x, y) \) is a Nash equilibrium of the bimatrix game \( \Gamma = (R, C) \) if and only if, for all \( k \in N \cup M \), \( x \in X(k) \) or \( y \in Y(k) \) (or both).

Now, since the game is nondegenerate, only finitely many strategies of the row player have \( n \) labels and only finitely many strategies of the column player have \( m \) labels. Using these finitely many strategies we define two graphs \( G_1 \) and \( G_2 \) as follows. Let \( G_1 \) be the graph whose nodes are the strategies \( x \in \mathbb{P}^n \) of the row player with exactly \( n \) labels, with an additional node \( 0 \in \mathbb{R}^n \) that has all labels in \( N \). There is an edge between any two nodes \( x, x' \) of \( G_1 \) if and only if \( x \) and
\( x' \) differ in exactly one label. Similarly, let \( G_2 \) be the graph whose nodes are the strategies \( y \in \mathbb{P}^m \) of the column player with exactly \( m \) labels, with an additional node \( 0 \in \mathbb{R}^m \) that has all labels in \( M \). There is an edge between any two nodes \( y, y' \) of \( G_2 \) if and only if \( y \) and \( y' \) differ in exactly one label.

Now define the product graph \( G_1 \times G_2 \) as the graph whose nodes are all pairs \((x, y)\) such that \( x \) is a node of \( G_1 \) and \( y \) is a node of \( G_2 \). There is an edge between any two nodes \((x, y), (x', y')\) if and only if (i) \( x = x' \) is a node of \( G_1 \) and \( (y, y') \) is an edge of \( G_2 \) or (ii) \( y = y' \) is a node of \( G_2 \) and \((x, x')\) is an edge of \( G_1 \).

For any \( k \in N \cup M \), call a node \((x, y)\) of \( G_1 \times G_2 \) \( k \)-almost completely labelled if any label \( \ell \in N \cup M \setminus \{k\} \) is a label of \( x \) or a label of \( y \). Since two adjacent nodes \( x \) and \( x' \) in \( G_1 \) have exactly \( n - 1 \) common labels, the edge \((x, y), (x', y')\) is also called \( k \)-almost completely labelled if \( y \) has the remaining \( m \) labels except \( k \). Similarly, since two adjacent nodes \( y \) and \( y' \) in \( G_2 \) have exactly \( m - 1 \) common labels, the edge \((x, y), (x, y')\) is called \( k \)-almost completely labelled if \( x \) has the remaining \( n \) labels except \( k \).

A Nash equilibrium \((x, y)\) is a node in \( G_1 \times G_2 \) adjacent to exactly one node \((x', y')\) that is \( k \)-almost completely labelled. In particular, if \( k \) is a label of \( x \), then \( x \) is joined to the node \( x' \) in \( G_1 \) sharing the remaining \( n - 1 \) labels, and \( y = y' \). If \( k \) is a label of \( y \), then \( y \) is joined to the node \( y' \) in \( G_2 \) sharing the remaining \( m - 1 \) labels, and \( x = x' \). Moreover, a \( k \)-almost completely labelled node \((x, y)\) that is completely labelled has exactly 2 neighbors in \( G_1 \times G_2 \). These are obtained by dropping the unique duplicate label that \( x \) and \( y \) have in common, joining to an adjacent node either in \( G_1 \) and keeping \( y \) fixed, or in \( G_2 \) and keeping \( x \) fixed. This defines a unique \( k \)-almost completely labelled path in \( G_1 \times G_2 \) connecting the completely labelled artificial equilibrium \((0, 0)\) to an actual equilibrium of \( \Gamma \).
The Lemke-Howson algorithm starts from \((0, 0)\), follows the path where label \(k\) is missing, and terminates at a Nash equilibrium of \(\Gamma\).

**Theorem 4.3 ([Lemke and Howson, 1964, Shapley, 1974])** Let \(\Gamma = \langle R, C \rangle\) be a nondegenerate bimatrix game and \(k \in N \cup M\). Then the set of \(k\)-almost completely labelled nodes and edges in \(G_1 \times G_2\) consists of disjoint paths and cycles. The endpoints of the paths are the equilibria of the game and the artificial equilibrium \((0, 0)\). The number of Nash equilibria of \(\Gamma\) is odd.

[Savani and von Stengel, 2004] proved that the Lemke-Howson may require an exponential number of steps for a specific class of inputs. Also note that the Lemke-Howson algorithm can be extended to degenerate bimatrix games as well, by “lexicographically perturbation” [von Stengel, 2002].

## 5 Approximate Equilibria

### 5.1 A quasi-polynomial algorithm

In [Althöfer, 1994] it is showed that, for any probability vector \(p\) there exists a probability vector \(\hat{p}\) with logarithmic supports, so that for a fixed matrix \(C\),

\[
\max_j |p^T C e_j - \hat{p}^T C e_j| \leq \epsilon,
\]

for any constant \(\epsilon > 0\). Exploiting this fact, Lipton et al., 2003 proved that, for any bimatrix game and for any constant \(\epsilon > 0\), there exists an \(\epsilon\)-Nash equilibrium with only logarithmic support (in the number \(n\) of available pure strategies). The proof is based on the probabilistic method and yields a quasi-polynomial algorithm for computing an \(\epsilon\)-Nash equilibrium for any \(\epsilon > 0\).
Theorem 5.1 ([Lipton et al., 2003]) For any Nash equilibrium \((\hat{x}, \hat{y})\) of an \(n \times n\) bimatrix game \(\Gamma = \langle R, C \rangle\) and for every \(\epsilon > 0\), there exists, for every \(k \geq \frac{12 \ln n}{\epsilon^2}\), a pair of \(k\)-uniform strategies \(x, y\), such that:

1. \((x, y)\) is an \(\epsilon\)-Nash equilibrium,
2. \(|x^T R y - \hat{x}^T R \hat{y}| < \epsilon\),
3. \(|x^T C y - \hat{x}^T C \hat{y}| < \epsilon\).

Proof: For the given \(\epsilon > 0\), fix some \(k \geq \frac{12 \ln n}{\epsilon^2}\). Form a multiset \(A\) by sampling \(k\) times from the set of actions of the row player, independently at random and according to the distribution \(\hat{x}\), and a multiset \(B\) by sampling \(k\) times from the set of actions of the column player, independently at random and according to the distribution \(\hat{y}\). Let \(x\) be the strategy for the row player that assigns probability \(\frac{1}{k}\) to each member of \(A\) and 0 each action not in \(A\). Similarly, let \(y\) be the strategy for the row player that assigns probability \(\frac{1}{k}\) to each member of \(B\) and 0 each action not in \(B\). Note that, if an action occurs \(\alpha\) times in the multiset, then it is assigned probability \(\frac{\alpha}{k}\).

Define the following events:

\[
\phi_1 = \{ |x^T R y - \hat{x}^T R \hat{y}| < \epsilon/2 \} \\
\pi_{1,i} = \{ e_i^T R y < x^T R y + \epsilon \} \quad \forall i \in \{1, \ldots, n\} \\
\phi_2 = \{ |x^T C y - \hat{x}^T C \hat{y}| < \epsilon/2 \} \\
\pi_{2,j} = \{ x^T C e_j < x^T C y + \epsilon \} \quad \forall j \in \{1, \ldots, n\} \\
\text{GOOD} = \phi_1 \cap \phi_2 \bigcap_{i=1}^n \pi_{1,i} \bigcap_{j=1}^n \pi_{2,j}.
\]
We wish to show that $\Pr\{\text{GOOD}\} > 0$, which would imply that there exists some choice of the multisets $A$ and $B$ such that $x$ and $y$ satisfy the three conditions in the statement of the theorem.

In order to bound the probabilities of the events $\pi_{1,i}$'s and $\pi_{2,j}$'s we introduce the following events:

$$
\psi_{1,i} = \{ e_i^T Ry < e_i^T R\hat{y} + \epsilon/2 \} \quad \forall i \in \{1, \ldots, n\} \quad (46)
$$

$$
\psi_{2,j} = \{ x^T Ce_j < \hat{x}^T C e_j + \epsilon/2 \} \quad \forall i \in \{1, \ldots, n\} \quad . \quad (47)
$$

Now assume that

$$
|x^T Ry - \hat{x}^T R\hat{y}| < \epsilon/2 \quad (48)
$$

or equivalently

$$
-\epsilon/2 < x^T Ry - \hat{x}^T R\hat{y} < \epsilon/2 \quad . \quad (49)
$$

Furthermore, assume that

$$
e_i^T Ry < e_i^T R\hat{y} + \epsilon/2 \quad (50)
$$

for some $i \in \{1, \ldots, n\}$. Then

$$
e_i^T Ry < e_i^T R\hat{y} + \epsilon/2 \quad (51)
$$

$$\leq \hat{x}^T R\hat{y} + \epsilon/2 \quad \text{(since ($\hat{x}, \hat{y}$) is a Nash equilibrium)} \quad (52)
$$

$$< x^T Ry + \epsilon \quad \text{(due to assumption (49))} \quad . \quad (53)
$$

Therefore,

$$
\psi_{1,i} \cap \phi_1 \subseteq \pi_{1,i} \quad \forall i \in \{1, \ldots, n\} \quad (54)
$$

and similarly it can be shown that

$$
\psi_{2,j} \cap \phi_2 \subseteq \pi_{2,j} \quad \forall j \in \{1, \ldots, n\} \quad . \quad (55)
$$
The expression $e^T R y$ is essentially a sum of $k$ independent random variables each of expected value $e_1 R \hat{y}$. Each such random variable takes values in the interval $[0, 1]$. Therefore we can apply a standard tail inequality [Hoeffding, 1963] and get:

$$\Pr\{\psi_{1,i}\} \leq \exp\left(-\frac{k\epsilon^2}{2}\right) \quad (56)$$

and similarly

$$\Pr\{\psi_{2,j}\} \leq \exp\left(-\frac{k\epsilon^2}{2}\right) \quad (57)$$

In order to bound the probabilities of the events $\phi_1^c$ and $\phi_2^c$ we define the following events:

$$\phi_{1a} = \{ |x^T R \hat{y} - \hat{x}^T R \hat{y}| < \epsilon/4 \} \quad (58)$$

$$\phi_{1b} = \{ |x^T R y - \hat{x}^T R \hat{y}| < \epsilon/4 \} \quad (59)$$

$$\phi_{2a} = \{ |\hat{x}^T C y - x^T C y| < \epsilon/4 \} \quad (60)$$

$$\phi_{2b} = \{ |x^T C y - \hat{x}^T C y| < \epsilon/4 \} \quad (61)$$

We can easily see that $\phi_{1a} \cap \phi_{1b} \subseteq \phi_1$ and $\phi_{2a} \cap \phi_{2b} \subseteq \phi_2$. The expression $x^T R \hat{y}$ is a sum of $k$ independent random variables, each of expected value $\hat{x}^T R \hat{y}$ and each taking values in the interval $[0, 1]$. Therefore we can apply the Hoeffding bound again and get:

$$\Pr\{\phi_{1a}^c\} \leq 2 \exp\left(-\frac{k\epsilon^2}{8}\right) \quad (62)$$

and, using similar arguments,

$$\Pr\{\phi_{1b}^c\} \leq 2 \exp\left(-\frac{k\epsilon^2}{8}\right) \quad (63)$$

Therefore

$$\Pr\{\phi_1^c\} \leq 4 \exp\left(-\frac{k\epsilon^2}{8}\right) \quad (64)$$
Similar reasoning yields that
\[ \Pr\{\phi'_c\} \leq 4 \exp\left( -\frac{k\epsilon^2}{8} \right). \tag{65} \]

Now
\[ \Pr\{\text{GOOD}^c\} \leq \Pr\{\phi'^c_1\} + \Pr\{\phi'^c_2\} + \sum_{i=1}^{n} \Pr\{\pi_{i,v_1}\} + \sum_{j=1}^{n} \Pr\{\pi_{j,v_2}\} \quad \tag{66} \]
\[ \leq \Pr\{\phi'^c_1\} + \Pr\{\phi'^c_2\} + \sum_{i=1}^{n} \left( \Pr\{\psi'_{i,v_1}\} + \Pr\{\phi'^c_1\} \right) \]
\[ + \sum_{j=1}^{n} \left( \Pr\{\psi'_{j,v_2}\} + \Pr\{\phi'^c_2\} \right) \quad \tag{67} \]
\[ \leq 8n \exp\left( -\frac{k\epsilon^2}{8} \right) + 8 \exp\left( -\frac{k\epsilon^2}{8} \right) + 2n \exp\left( -\frac{k\epsilon^2}{2} \right) \quad \tag{68} \]
\[ < 1 \quad (\text{since } k \geq 12 \ln n/\epsilon^2), \quad \tag{69} \]
and therefore \( \Pr\{\text{GOOD}\} > 0 \) as needed. $\square$

**Corollary 5.2 ([Lipton et al., 2003])** For an \( n \times n \) bimatrix game \( \Gamma = \langle R, C \rangle \), there exists a quasi-polynomial algorithm for computing all \( k \)-uniform \( \epsilon \)-Nash equilibria, for any \( \epsilon > 0 \).

**Proof:** For the given \( \epsilon > 0 \), fix \( k = \frac{12 \ln n}{\epsilon^2} \). By exhaustive search, we can find all possible pairs of multisets \( A \) and \( B \). For each such pair, we can check in polynomial time if the pair of \( k \)-uniform strategies is an \( \epsilon \)-Nash equilibrium. By Theorem 5.1, at least one pair of multisets exists such that the corresponding \( k \)-uniform strategies are \( \epsilon \)-Nash equilibrium strategies. The running time of the algorithm is quasi-polynomial, since there are \( \left( \frac{n+k-1}{k} \right)^2 \) possible pairs of multisets to check. $\square$

As pointed out in [Althöfer, 1994], no algorithm that examines supports smaller than about \( \ln n \) can achieve an approximation better than \( \frac{1}{4} \). Moreover, [Chen et al., 2006] proved the following:
Theorem 5.3 ([Chen et al., 2006]) The problem of computing a $\frac{1}{\Theta(n)}$-Nash equilibrium of a $n \times n$ bimatrix game is PPAD-complete.

Theorem 5.3 asserts that, unless PPAD $\subseteq$ P, there exists no fully polynomial time approximation scheme for computing equilibria in bimatrix games. However, this does not rule out the existence of a polynomial approximation scheme for computing an $\epsilon$-Nash equilibrium when $\epsilon$ is an absolute constant, or even when $\epsilon = \Theta\left(\frac{1}{\text{poly}(\ln n)}\right)$. Furthermore, as observed in [Chen et al., 2006], if the problem of finding an $\epsilon$-Nash equilibrium were PPAD-complete when $\epsilon$ is an absolute constant, then, due to Theorem 5.1, all PPAD problems would be solved in quasi-polynomial time, which is unlikely to be the case.

Two concurrent and independent works [Daskalakis et al., 2006b; Kontogiannis et al., 2006] were the first to make progress in providing $\epsilon$-Nash equilibria for bimatrix games and some constant $0 < \epsilon < 1$. In particular, the work of [Kontogiannis et al., 2006] proposes a simple linear-time algorithm for computing a $\frac{3}{4}$-Nash equilibrium for any bimatrix game:

**Theorem 5.4 ([Kontogiannis et al., 2006])** Consider any $n \times m$ bimatrix game $\Gamma = \langle R, C \rangle$ and let $r_{i_1,j_1} = \max_{i,j} r_{ij}$ and $c_{i_2,j_2} = \max_{i,j} c_{ij}$. Then the pair of strategies $(\hat{x}, \hat{y})$ where $\hat{x}_{i_1} = \hat{x}_{i_2} = \hat{y}_{j_1} = \hat{y}_{j_2} = \frac{1}{2}$ is a $\frac{3}{4}$-Nash equilibrium for $\Gamma$.

The above technique can be extended so as to obtain a parameterized, stronger approximation:

**Theorem 5.5 ([Kontogiannis et al., 2006])** Consider a $n \times m$ bimatrix game $\Gamma = \langle A, B \rangle$. Let $\lambda_1^* (\lambda_2^*)$ be the minimum, among all Nash equilibria of $\Gamma$, expected payoff for the row (column) player and let $\lambda = \max\{\lambda_1^*, \lambda_2^*\}$. Then, there exists a $\frac{2+\lambda}{4}$-Nash equilibrium that can be computed in time polynomial in $n$ and $m$. 
The work of [Daskalakis et al., 2006b] provides a simple algorithm for computing a \( \frac{1}{2} \)-Nash equilibrium: Pick an arbitrary row for the row player, say row \( i \). Let \( j = \arg \max \limits_{j'} c_{ij'} \). Let \( k = \arg \max \limits_{k'} r_{kj} \). Thus, \( j \) is a best-response column for the column player to the row \( i \), and \( k \) is a best-response row for the row player to the column \( j \). Let \( \hat{x} = \frac{1}{2}e_i + \frac{1}{2}e_k \) and \( \hat{y} = e_j \), i.e., the row player plays row \( i \) or row \( k \) with probability \( \frac{1}{2} \) each, while the column player plays column \( j \) with probability 1. Then:

**Theorem 5.6 (Daskalakis et al., 2006b)** The strategy profile \((\hat{x}, \hat{y})\) is a \( \frac{1}{2} \)-Nash equilibrium.

A polynomial construction (based on Linear Programming) of a 0.38-Nash equilibrium is presented in [Daskalakis et al., 2007].

For the more demanding notion of well-supported approximate Nash equilibrium, [Daskalakis et al., 2006b] propose an algorithm, which, under a quite interesting and plausible graph theoretic conjecture, constructs in polynomial time a \( \frac{5}{6} \)-well-supported Nash equilibrium. However, the status of this conjecture is still unknown. In [Daskalakis et al., 2006b] it is also shown how to transform a \([0,1]\)-bimatrix game to a win-lose bimatrix game (i.e. a bimatrix game where each matrix entry is either 0 or 1) of the same size, so that each \( \epsilon \)-well supported Nash equilibrium of the resulting game is \( \frac{1+\epsilon}{2} \)-well supported Nash equilibrium of the original game.

The work of [Kontogiannis and Spirakis, 2007] provides a polynomial algorithm that computes a \( \frac{1}{2} \)-well-supported Nash equilibrium for arbitrary win-lose games. The idea behind this algorithm is to split evenly the divergence from a zero-sum game between the two players and then solve this zero-sum game in
polynomial time (using its direct connection to Linear Programming). The computed Nash equilibrium of the zero-sum game considered is indeed proved to be also a $\frac{1}{2}$-well-supported Nash equilibrium for the initial win-lose game. Therefore:

**Theorem 5.7 ([Kontogiannis and Spirakis, 2007])**  For any win-lose bimatrix game, there is a polynomial time constructible profile that is a $\frac{1}{2}$-well-supported Nash equilibrium of the game.

In the same work, [Kontogiannis and Spirakis, 2007] parameterize the above methodology in order to apply it to arbitrary bimatrix games. This new technique leads to a weaker $\phi$-well-supported Nash equilibrium for win-lose games, where $\phi = \frac{\sqrt{5} - 1}{2}$ is the golden ratio. Nevertheless, this parameterized technique extends nicely to a technique for arbitrary bimatrix games, which assures a $\left(\frac{\sqrt{11}}{2} - 1\right)$-well-supported Nash equilibrium in polynomial time:

**Theorem 5.8 (Kontogiannis and Spirakis, 2007)**  For any bimatrix game, a 0.658-well-supported Nash equilibrium is constructible in polynomial time.

## 6 Potential Games and Congestion Games

### 6.1 Potential Games

Potential games, defined in [Monderer and Shapley, 1996], are games with the property that the incentive of all players to unilaterally deviate from a pure strategy profile can be expressed in one global function, the potential function.

Fix an arbitrary game in strategic form $\Gamma = (N, (C_i)_{i \in N}, (u_i)_{i \in N})$ and some vector $b \in \mathbb{R}^{|N|}$. A function $P : C \to \mathbb{R}$ is called
• An ordinal potential for $\Gamma$ if, $\forall c \in C$, $\forall i \in N$, $\forall a \in C_i$,
\[
P(a, c_{-i}) - P(c) > 0 \iff u_i(a, c_{-i}) - u_i(c) > 0.
\] (70)

• A $b$-potential for $\Gamma$ if, $\forall c \in C$, $\forall i \in N$, $\forall a \in C_i$,
\[
P(a, c_{-i}) - P(c) = b_i \cdot (u_i(a, c_{-i}) - u_i(c)).
\] (71)

• An exact potential for $\Gamma$ if it is a $b$-potential for $\Gamma$ where $b_i = 1$ for all $i \in N$.

It is straightforward to see that the existence of an ordinal, exact, or $b$-potential function $P$ for a finite game $\Gamma$ guarantees the existence of pure Nash equilibria in $\Gamma$: each local optimum of $P$ corresponds to a pure Nash equilibrium of $\Gamma$ and vice versa. Thus the problem of finding pure Nash equilibria of a potential game $\Gamma$ is equivalent to finding local optima for the optimization problem with state space the pure strategy space $C$ of the game and objective the potential function of the game.

Furthermore, the existence of a potential function $P$ for a game $\Gamma = \langle N, (C_i)_{i \in N}, (u_i)_{i \in N} \rangle$ implies a straightforward algorithm for constructing a pure Nash equilibrium of $\Gamma$: The algorithm starts from an arbitrary strategy profile $c \in C$ and, at each step, one single player performs a selfish step, i.e. switches to a pure strategy that strictly improves her payoff. Since the payoff of the player increases, $P$ increases as well. When no move is possible, i.e. when a pure strategy profile $\hat{c}$ is reached from which no player has an incentive to unilaterally deviate, then $\hat{c}$ is a pure Nash equilibrium and a local optimum of $P$. This procedure however does not imply that the computation of a pure Nash equilibrium can be done in polynomial time, since the improvements in the potential can be very small and too many.
6.2 Congestion Games

[Rosenthal, 1973] introduced a class of games, called congestion games, in which each player chooses a particular subset of resources out of a family of allowable subsets for her (her action set), constructed from a basic set of primary resources for all the players. The delay associated with each primary resource is a non-decreasing function of the number of players who choose it, and the total delay received by each player is the sum of the delays associated with the primary resources she chooses. Each game in this class possesses at least one Nash equilibrium in pure strategies, which follows from the existence of an exact potential.

A congestion model \( \langle N, E, (\Pi_i)_{i \in N}, (d_e)_{e \in E} \rangle \) is defined as follows. \( N \) denotes the set of players \( \{1, \ldots, n\} \). \( E \) denotes a finite set of resources. For \( i \in N \) let \( \Pi_i \) be the set of strategies of player \( i \), where each \( \varpi_i \in \Pi_i \) is a nonempty subset of resources. For \( e \in E \) let \( d_e : \{1, \ldots, n\} \to \mathbb{R} \) denote the delay function, where \( d_e(k) \) denotes the cost (e.g. delay) to each user of resource \( e \), if there are exactly \( k \) players using \( e \).

The congestion game associated with this congestion model is the game in strategic form \( \langle N, (\Pi_i)_{i \in N}, (u_i)_{i \in N} \rangle \), where the payoff functions \( u_i \) are defined as follows: Let \( \Pi \equiv \times_{i \in N} \Pi_i \). For all \( \varpi = (\varpi_1, \ldots, \varpi_n) \in \Pi \) and for every \( e \in E \) let \( \sigma_e(\varpi) \) be the number of users of resource \( e \) according to the configuration \( \varpi \): \( \sigma_e(\varpi) = |\{i \in N : e \in \varpi_i\}| \). Define \( u_i : \Pi \to \mathbb{R} \) by \( u_i(\varpi) = -\sum_{e \in \varpi_i} d_e(\sigma_e(\varpi)) \).

In a network congestion game the families of subsets \( \Pi_i \) are represented implicitly as paths in a network. We are given a directed network \( G = (V, E) \) with the edges playing the role of resources, a pair of nodes \( (s_i, t_i) \in V \times V \) for each player \( i \) and the delay function \( d_e \) for each \( e \in E \). The strategy set of player \( i \) is
the set of all paths from $s_i$ to $t_i$. If all origin-destination pairs $(s_i, t_i)$ of the players coincide with a unique pair $(s, t)$ we have a single-commodity network congestion game and then all users share the same strategy set, hence the game is symmetric.

In a weighted congestion model we allow the users to have different demands, and thus affect the resource delay functions in a different way, depending on their own weights. A weighted congestion model $\langle N, (w_i)_{i \in N}, (\Pi_i)_{i \in N}, (d_e)_{e \in E} \rangle$ is defined as follows. $N$ denotes the set of players $\{1, 2, \ldots, n\}$, $w_i$ denotes the demand of player $i$, and $E$ denotes a finite set of resources. For $i \in N$ let $\Pi_i$ be the set of strategies of player $i$, where each $\varpi_i \in \Pi_i$ is a nonempty subset of resources. For each resource $e \in E$ let $d_e(\cdot)$ be the delay per user that requests its service, as a function of the total usage of this resource by all the users.

The weighted congestion game associated with this congestion model is the game in strategic form $\langle (w_i)_{i \in N}, (\Pi_i)_{i \in N}, (u_i)_{i \in N} \rangle$, where the payoff functions $u_i$ are defined as follows. For any configuration $\varpi \in \Pi$ and for all $e \in E$, let $\Lambda_e(\varpi) = \{i \in N : e \in \varpi_i\}$ be the set of players using resource $e$ according to $\varpi$. The cost $\lambda^i(\varpi)$ of user $i$ for adopting strategy $\varpi_i \in \Pi_i$ in a given configuration $\varpi$ is equal to the cumulative delay $\lambda_{\varpi_i}(\varpi)$ on the resources that belong to $\varpi_i$:

$$\lambda^i(\varpi) = \lambda_{\varpi_i}(\varpi) = \sum_{e \in \varpi_i} d_e(\theta_e(\varpi))$$

(72)

where, for all $e \in E$, $\theta_e(\varpi) = \sum_{i \in \Lambda_e(\varpi)} w_i$ is the load on resource $e$ with respect to the configuration $\varpi$. The payoff function for player $i$ is then $u_i(\varpi) = -\lambda^i(\varpi)$.

A configuration $\varpi \in \Pi$ is a pure Nash equilibrium if and only if, for all $i \in N$,

$$\lambda_{\varpi_i}(\varpi) \leq \lambda_{\pi_i}(\pi_i, \varpi_{-i}) \quad \forall \pi_i \in \Pi_i$$

(73)

where $(\pi_i, \varpi_{-i})$ is the same configuration as $\varpi$ except for user $i$ that has now been assigned to path $\pi_i$. 

30
In a weighted network congestion game the strategy sets \( \Pi_i \) are represented implicitly as \( s_i - t_i \) paths in a directed network \( G = (V, E) \).

Since the payoff functions \( u_i \) of a congestion game can be implicitly computed by the resource delay functions \( d_e \), in the following we will denote a general (weighted or unweighted) congestion game by \( \langle N, E, (\Pi_i)_{i \in N}, (w_i)_{i \in N}, (d_e)_{e \in E} \rangle \). We drop \((w_i)_{i \in N}\) from this notation when referring to an unweighted congestion game.

The following theorem [Rosenthal, 1973, Monderer and Shapley, 1996] proves the strong connection of unweighted congestion games with the exact potential games.

**Theorem 6.1 ([Rosenthal, 1973, Monderer and Shapley, 1996])** Every (unweighted) congestion game is an exact potential game.

**Proof:** Fix an arbitrary (unweighted) congestion game \( \Gamma = \langle N, E, (\Pi_i)_{i \in N}, (d_e)_{e \in E} \rangle \). For any pure strategy profile \( \varpi \in \Pi \), the function

\[
\Phi(\varpi) = \sum_{e \in \bigcup_{i \in N} \varpi_i} \sum_{k=1}^{\sigma_e(\varpi)} d_e(k)
\]

(introduced in [Rosenthal, 1973]) is an exact potential function for \( \Gamma \). \( \square \)

The converse of Theorem 6.1 does not hold in general, however [Monderer and Shapley, 1996] proved that every (finite) exact potential game \( \Gamma \) is isomorphic to an unweighted congestion game.

### 6.2.1 The Complexity Class PLS

A crucial class of problems containing the family of weighted congestion games is PLS (standing for Polynomial Local Search) [Johnson et al., 1988]. This is the
subclass of total functions in \( \text{NP} \) that are guaranteed to have a solution because of the fact that “every finite directed acyclic graph has a sink”.

A local search problem \( \Pi \) has a set of instances \( D_\Pi \) which are strings. To each instance \( x \in D_\Pi \), there corresponds a set \( S_\Pi(x) \) of solutions, and a standard solution \( s_0 \in S_\Pi(x) \). Each solution \( s \in S_\Pi(x) \) has a cost \( f_\Pi(s, x) \) and a neighborhood \( N_\Pi(s, x) \). The search problem is, given an instance \( x \in D_\Pi \), to find a locally optimal solution \( s^* \in S_\Pi(x) \). That is, if the objective is to minimize the cost function, then we are seeking for some \( s^* \in \arg\min_{s \in N_\Pi(s^*, x)} \{ f_\Pi(s, x) \} \). Similarly, if the objective is to maximize the cost function, then we are seeking for some \( s^* \in \arg\max_{s \in N_\Pi(s^*, x)} \{ f_\Pi(s, x) \} \).

**Definition 6.1** A local search problem \( \Pi \) is in \( \text{PLS} \) if its instances \( D_\Pi \) and solutions \( S_\Pi(x) \) for all \( x \in D_\Pi \) are binary strings, there is a polynomial \( p \) such that the length of the solutions \( S_\Pi(x) \) is bounded by \( p(|x|) \), and there are three polynomial-time algorithms \( A_\Pi, B_\Pi, C_\Pi \) with the following properties:

1. Given a binary string \( x \), \( A_\Pi \) determines whether \( x \in D_\Pi \), and if so, returns some initial solution \( s_0 \in S_\Pi(x) \).

2. Given an instance \( x \in D_\Pi \) and a string \( s \), \( B_\Pi \) determines whether \( s \in S_\Pi(x) \), and if so, computes the value \( f_\Pi(s, x) \) of the cost function at \( s \).

3. Given an instance \( x \in D_\Pi \) and a solution \( s \in S_\Pi(x) \), \( C_\Pi \) determines whether \( s \) is a local optimum of \( f_\Pi(\cdot, x) \) in its neighborhood \( N_\Pi(s, x) \), and if not, returns a neighbor \( s' \in N_\Pi(s, x) \) having a better value (i.e. \( f_\Pi(s', x) < f_\Pi(s, x) \) for a minimization problem and \( f_\Pi(s', x) > f_\Pi(s, x) \) for a maximization problem).
6.2.2 Complexity of Pure Nash Equilibria in Congestion Games

[Fabrikant et al., 2004] characterized the complexity of computing pure Nash equilibria in congestion games. They showed that, for unweighted single commodity network congestion games, a pure Nash equilibrium can be constructed in polynomial time. On the other hand, they showed that, even for symmetric congestion games it is PLS-complete to compute a pure Nash equilibrium.

**Theorem 6.2 ([Fabrikant et al., 2004])** There is a polynomial time algorithm for finding a pure Nash equilibrium in symmetric network congestion games.

**Proof:** The algorithm is a reduction to min-cost flow. Given the single-source single-destination network \( G = (V, E) \) and the delay functions \( d_e \), we replace each edge \( e \) with \( n \) parallel edges between the same nodes, each with capacity 1, and with costs \( d_e(1), \ldots, d_e(n) \). It is straightforward to see that any min-cost flow in the resulting network (which can be computed in polynomial time) is integral and it corresponds to a pure strategy profile of the congestion game that minimizes its exact potential function, thus it corresponds to a pure Nash equilibrium. \( \square \)

**Theorem 6.3 ([Fabrikant et al., 2004])** It is PLS-complete to find a pure Nash equilibrium in unweighted congestion games of the following sorts:

(i) General congestion games.

(ii) Symmetric congestion games.

(iii) Multi-commodity network congestion games.

**Proof:** In order to prove (i) we shall use the following problems:
**NOTALLEQUAL3SAT**: Given a set $N$ of binary variables and a collection $C$ of clauses such that $\forall c \in C, |c| \leq 3$, is there an assignment of values to the variables so that no clause has all its literals assigned the same value?

**POSNAE3FLIP**: Given an instance $(N, C)$ of NOTALLEQUAL3SAT with positive literals only and a weight function $w : C \rightarrow \mathbb{R}$, find an assignment of values to the variables such that the total weight of the unsatisfied clauses and the totally satisfied (i.e. with all their literals set to 1) can not be decreased by a unilateral flip of the value of any variable.

POSNAE3FLIP is known to be PLS-complete. Given an instance of POSNAE3FLIP, we construct a congestion game as follows. The set of players is exactly the set of variables. For each 3-clause $c$ of weight $w_c$ we have two resources $e_c$ and $e'_c$, with delay 0 if there are two or fewer players, and $w_c$ otherwise. Player $v$ has two actions: one action contains all resources $e_c$ for clauses that contain $v$, and one action that contains all resources $e'_c$ for the same clauses. Smaller clauses are implemented similarly. Clearly, a flip of a variable corresponds to the change in the pure strategy of the corresponding player. The changes in the total weight due to a flip equal the changes in the cumulative delay over all the resources. Thus, any pure Nash equilibrium of the congestion game is a local optimum (and therefore a solution) of the POSNAE3FLIP problem, and vice versa.

The proof of (ii) is by reduction of the non-symmetric case to the symmetric case. Given an unweighted congestion game $\Gamma = \langle N, E, (\Pi_i)_{i \in N}, (d_e)_{e \in E} \rangle$, we construct a symmetric congestion game $\hat{\Gamma} = \langle N, E, \hat{\Pi}, (\hat{d}_e)_{e \in E} \rangle$ as follows. First we add to the set of resources $n$ distinct resources: $\hat{E} = E \cup \{e_i\}_{i \in N}$. The delays of these resources are $\hat{d}_{e_i}(k) = M$ for some sufficiently large constant $M$ if $k \geq 2$, and 0 if $k = 1$. The old resources maintain the same delay functions. Each player
has the same action set \( \hat{\Pi} = \bigcup_{i \in N} \{ \pi_i \cup \{ e_i \} : \pi_i \in \Pi_i \} \). By setting the constant \( M \) sufficiently large, in any pure Nash equilibrium of \( \Gamma \) each of the distinct resources is used by exactly one player (these resources act as if they have capacity 1 and there are only \( n \) of them). Therefore, in any pure Nash equilibrium, each one of the \( n \) players chooses a different subset of resources. So, for any pure Nash equilibrium in \( \hat{\Gamma} \) we can easily get a pure Nash equilibrium in \( \Gamma \) by simply dropping the unique new resource used by each of the players. This is done by identifying the “anonymous” players of \( \hat{\Gamma} \) according to the unique resource they use, and match them with the corresponding players of \( \Gamma \).

The proof of (iii) is rather complicated and will not be presented here; we refer the interested reader to [Fabrikant et al., 2004]. □

Next we deal with the existence and tractability of pure Nash equilibria in weighted network congestion games. In [Fotakis et al., 2005] it was shown that it is not always the case that a pure Nash equilibrium exists:

**Theorem 6.4 ([Fotakis et al., 2005])** There exist instances of weighted single-commodity network congestion games with resource delays being either linear or 2-wise linear function of the loads, for which there is no pure Nash equilibrium.

Furthermore, it was shown that there may exist no exact potential function even for the simplest case of weighted congestion games:

**Theorem 6.5 ([Fotakis et al., 2005])** There exist weighted single-commodity network congestion games which are not exact potential games, even when the resource delays are identical to their loads.

This fact however does not rule out the possibility that there exists some non-exact potential for a weighted network congestion game. Actually, [Fotakis et al., 2005]
proved the existence of a b-potential function for any weighted multi-commodity network congestion game with linear resource delays:

**Theorem 6.6 ([Fotakis et al., 2005])** For any weighted multi-commodity network congestion game with linear resource delays (i.e. \( d_e(x) = a_e x + c_e, e \in E, a_e, c_e \geq 0 \)), at least one pure Nash equilibrium exists and can be computed in pseudo-polynomial time.

**Proof:** The result follows from the existence of the b-potential function

\[
\Phi(\varpi) = \sum_{e \in E} \left( a_e \theta_e^2(\varpi) + c_e \theta_e(\varpi) + \sum_{i: \varpi_i = e} (a_i w_i^2 + c_i w_i) \right)
\]

where \( b_i = \frac{1}{\tau w_i} \) for all \( i \in N \).

\[ \Box \]

7 The Price of Anarchy

In a game-like situation, the *price of anarchy* (or coordination ratio), introduced in [Koutsoupias and Papadimitriou, 1999], is a measure of the performance degradation due to the selfish behavior of the involved players (and therefore due to the lack of coordination among the players). In general, it is defined with respect to a global *social cost* function, capturing the performance of a strategy profile. Then the price of anarchy equals the ratio of the worst social cost over all Nash equilibria and the optimum value of the social cost.

In the following, we will focus on the characterization of the price of anarchy in the context of network congestion games. The social cost in this setting can be defined in several ways, such as the maximum congestion over all edges of the network, the average congestion of the edges, the sum of the costs over all players.
etc. In the following, we define the social cost as the maximum cost paid over all agents.

Formally, consider a weighted network congestion game \( \Gamma = (N, E, (\Pi_i)_{i \in N}, (w_i)_{i \in N}, (d_e)_{e \in E}) \) and let \( G = (V, E) \) be the corresponding network. Let \( p = (p_i^j)_{i \in N, j \in \Pi_i} \) be an arbitrary strategy profile, that is, \( p_i^j \) is the probability that player \( i \in N \) chooses action \( j \in \Pi_i \). Then the social cost in this congestion game is

\[
SC(p) = \sum_{\varpi \in \Pi} \Pr(p, \varpi) \cdot \max_{i \in N} \{\lambda_{\varpi_i}(\varpi)\} \tag{76}
\]

where \( \Pr(p, \varpi) = \prod_{i \in N} p_i^{\varpi_i} \) is the probability of pure strategy profile \( \varpi \) occurring, with respect to the strategy profile \( p \).

The social optimum of this game is defined as

\[
OPT = \min_{\varpi \in \Pi} \left\{ \max_{i \in N} \{\lambda_{\varpi_i}(\varpi)\} \right\}. \tag{77}
\]

The price of anarchy is then defined as the ratio of the social cost of the worst Nash equilibrium and \( OPT \):

\[
R = \max_{p \text{ is a NE}} \frac{SC(p)}{OPT}. \tag{78}
\]

In the seminal paper [Koutsoupias and Papadimitriou, 1999] it was proved that the price of anarchy for the special case of a network consisting of 2 parallel links is \( R = 3/2 \), while for \( m \) parallel links, \( R = \Omega \left( \frac{\log m}{\log \log m} \right) \) and \( R = O \left( \sqrt{m \log m} \right) \).

For \( m \) identical parallel links, [Mavronicolas and Spirakis, 2001] proved that \( R = \Theta \left( \frac{\log m}{\log \log m} \right) \) for the fully mixed Nash equilibrium, i.e. for the Nash equilibrium where each player poses nonzero probability to each link. For the case of \( m \) identical parallel links it was shown in [Koutsoupias et al., 2003] that \( R = \Theta \left( \frac{\log m}{\log \log m} \right) \). In [Czumaj and Vöcking, 2002] it was finally shown that \( R = \Theta \left( \frac{\log m}{\log \log m} \right) \) for the general case of non-identical parallel links (and players of varying demands).
In [Roughdarden and Tardos, 2002] the price of anarchy in a multicommodity network congestion game among infinitely many players, each of negligible demand, is studied. The social cost in this case is expressed by the total delay paid by the whole flow in the system. For linear resource delays, the price of anarchy is at most 4/3. For general, continuous, non-decreasing resource delay functions, the total delay of any Nash flow is at most equal to the total delay of an optimal flow for double flow demands. [Roughgarden, 2002] proves that for this setting, it is actually the class of allowable latency functions and not the specific topology of a network that determines the price of anarchy.

As pointed out in [Fotakis et al., 2005], there exist weighted network congestion games with resource delays linear to their loads for which the price of anarchy is unbounded. This result holds even for the simple case of an $\ell$-layered network, i.e. a single-commodity network $(V, E)$ where every source-destination path has length exactly $\ell$ and each node lies on a directed source-destination path. However, if resource delays are identical to their loads, the following holds:

**Theorem 7.1 ([Fotakis et al., 2005])** The price of anarchy of any single-commodity network congestion game with resource delays identical to their loads, is at most

$$8e \left( \frac{m}{\log \log m} + 1 \right),$$

where $m = |E|$ is the number of available resources, i.e. the number of edges of the network.

[Christodoulou and Koutsoupias, 2005] considered the price of anarchy of pure Nash equilibria in congestion games with linear latency functions. For asymmetric games, they showed that the price of anarchy when the social cost is defined as the
maximum congestion over all resources is $\Theta(\sqrt{n})$ where $n$ is the number of players. For all other cases of symmetric or asymmetric games and for both maximum and average social cost, the price of anarchy was shown to be $5/2$, and these results were extended to latency functions that are polynomials of bounded degree.

[Gairing et al., 2006] studied an interesting variant of the price of anarchy in weighted congestion games on parallel links, where the social cost is the expectation of the sum, over all links, of latency costs; each latency cost is modeled as a certain polynomial function evaluated at the delay incurred by all users choosing the link.

The main argument for using worst-case demands in the definition of the price of anarchy is that the distribution of the players’ demands is not known, and the worst-case distribution prevailing in this definition is the one in which the worst-case demand occurs with probability one. [Mavronicolas et al., 2005] introduced the notion of Diffuse Price of Anarchy in order to remove this assumption while avoiding to assume full knowledge about the distribution of demands. Roughly speaking, the Diffuse Price of Anarchy is the worst-case, over all allowed probability distributions, of the expectation (according to each specific probability distribution) of the ratio of social cost over optimum in the worst-case Nash equilibrium.

8 Defining Terms

**Finite strategic form game:** A finite strategic form game is defined by a nonempty finite set of players, and, for each player, a nonempty finite set of actions and a payoff function mapping each combination of actions (one for each player) to a real number.
**Mixed strategy:** A mixed strategy for a player is any probability distribution on her set of actions.

**Pure strategy:** A pure strategy for a player is a mixed strategy that poses probability 1 to exactly one of her actions, and 0 to all the rest.

**Strategy profile:** A strategy profile is a combination of (pure or mixed) strategies, one for each player.

**Support:** The support of a strategy of some player is the subset of actions that are assigned nonzero probability.

**Nash equilibrium:** A strategy profile is a Nash equilibrium if no player can increase her payoff by unilaterally deviating from the profile.

**Bimatrix game:** A bimatrix game is a 2-player finite strategic form game, such that the players’ payoffs can be described by two real matrices $R$ and $C$, where each row corresponds to an action of the first player and each column corresponds to an action of the second player. The payoffs when the first player chooses her $i$th action and the second player chooses her $j$th action are given by the $(i,j)$th element of matrix $R$ for the first player, and by the $(i,j)$th element of matrix $C$ for the second player.

**$\epsilon$-Nash equilibrium:** A strategy profile is an $\epsilon$-Nash equilibrium if no player can increase her payoff by more than $\epsilon$ by unilaterally deviating from the profile.

**$\epsilon$-well-supported Nash equilibrium:** A strategy profile is an $\epsilon$-well-supported Nash equilibrium if it is an $\epsilon$-Nash equilibrium with the additional property that each player plays with nonzero probability only those actions that guarantee her a payoff no less than the payoff that the specific strategy profile gives her, plus $\epsilon$.

**Price of anarchy:** In a game-like situation, the price of anarchy (or coordination ratio) is a measure of the performance degradation due to the selfish behavior of the
involved players (and therefore due to the lack of coordination among the players). In general, it is defined with respect to a global social cost function, capturing the performance of a strategy profile. Then the price of anarchy equals the ratio of the worst social cost over all Nash equilibria and the optimum value of the social cost.

References


X. Chen and X. Deng. 3-Nash is PPAD-complete. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.


