ALGORITHMIC GAME THEORY COLUMN

BY

MARIO MAVRONICOLAS

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

A STATE-OF-THE-ART CATALOG OF COMPLEXITY CLASSES WITH PROBLEMS OF EQUILIBRIUM COMPUTATION *

Vicky Papadopoulou † Marios Mavronicolas

Abstract

The problem of computing equilibria is arguably one of the most important computational problems in Algorithmic Game Theory today. The complexity of this problem has been found to be very sensitive on the definition of equilibrium used, the class of games considered and additional requirements on the solution.

In this work, we provide a glimpse at some of the most exciting recent developments around this problem. Perhaps unusually, we follow a horizontal approach: we categorize the developments by complexity class; hence, we provide a state-of-the-art catalog of complexity classes that have been enriched with problems of equilibrium computation.

1 Introduction

Noncooperative Game Theory has been providing the dominating normative framework for analyzing strategic interactions of agents. In order the framework to be operational, the solutions it defines will have to be *efficiently computable*. According to Kamal Jain's famous quote, "if your laptop cannot compute, the network cannot either".

- Game Theory is scaled to large detailed real-world settings. Thus, it will become necessary to use computers to solve the games. Computational complexity will then determine the scalability of the game-theoretic toolbox.

*This is an extended version of Deliverable 4.1.3 for DELIS, a project supported by the European Union under contract number IST-2004-001907.

†Department of Computer Science & Engineering, European University Cyprus, 6, Diogenes Str., Engomi, P.O. Box: 22006, 1516 Nicosia-Cyprus.

- Algorithms for solving games consists a necessary component of sophisticated agents that are to act in games. Computational complexity will then affect the scalability and usability of such agents.

- Computational complexity can be used to predict the descriptive power of a solution concept. If a solution is too hard to find, it is unlikely that the players will act according to the solution concept.

- The computational complexity of a solution concept gives a lower bound on the complexity of any learning algorithm for finding a solution according to the concept.

- Computational complexity of solving a game plays roles in mechanism design as well. On the negative side, if solving a game is too hard for an agent, the agent is likely not going to play as the designer incentivized the agent to play. In many settings it is possible to achieve socially more desirable outcomes by making the agents face prohibitive computational complexity in solving the game [28, 8].

There has been growing interest in the computational complexity of natural questions in game theory. Starting at least as early as the 1970s, complexity theorists have focused on the complexity of playing particular highly structured games (usually board games, such as chess or Go [36], but also games such as Geography or QSAT [53]. These games tend to be alternating-move zero-sum games with enormous state spaces, which can nevertheless be concisely represented due to the simple rules governing the transition between states. As a result, effort on finding results for general classes of games has often focused on complex languages in which such structured games can be concisely represented. Real-world strategic settings are generally not nearly as structured, nor do they generally possess the other properties of board games. So, understanding the complexity of solving these broader games is critical.

Games are commonly represented in two forms: strategic/normal, where the game takes the form of a matrix of payoffs, and extensive, where the game takes the form of a tree where each layer of the tree is one playerŠs move. In this survey we concentrate on games described in a normal form:

Definition 1.1. *A* normal form game *is a collection $S_1, \cdots, S_n$ of finite strategy sets and a collection $u_1, \cdots, u_n$ of utility functions, each defined on $S_1 \times \cdots \times S_n$. We identify a strategy set S_i and utility function u_i with player i. A element s of $S_1 \times \cdots \times S_n$ is called a strategy profile.*

A game with n players is called n-player game.

An algorithm for a game in *extensive form* could be applied to a game in normal form since the normal form matrix could be transformed to extensive form easily; the reverse is not true since according to Gilboa's result [21], the transformation from extensive form into normal form is already exponential. Thus, there are separate algorithms for games in the two forms, and the complexity results for the two forms may be different because matrices are not transformed easily between the two forms.

In this deliverable, we provide a a state-of-the-art catalog of those complexity classes that have been enriched with problems of computing network equilibria and rational behavior of agents.

Site Map In subsection 1.1, we provide necessary definitions of the most important solution concepts on Game Theory; of which Nash equilibrium stands as a primer notion. In section 2, we briefly describe the most important computational complexity classes that have been enriched with problems of computing Nash equilibria and rational behavior of agents. In section 3.1, we provide simple but sufficient definitions of some of the most important games, that have been investigated the last decades, and we will present next various results. Next, we present, a comprehensive catalog of problems and the corresponding solutions that have been presented. We categorized the problems according to their proven computational complexity. Thus, we overview on a great variance of problems that have been shown to belong or to be complete in one of the most important computational classes including, P, $\mathcal{NP}$, $\mathcal{PLS}$, $\mathcal{PPAD}$, $\mathcal{NEXP}$, $\#\mathsf{P}$, Σ_k^P and Polynomial Space. We devote a section to each one of these computational classes.

1.1 Solution Concepts

1.1.1 Nash Equilibria

Game theory provides languages to represent such strategic settings as well as notions of what it means to "solve" them. The *Nash equilibrium* [41, 42] is a solution concept, defined as the strategy of best response for each player. That is, the Nash equilibrium is the strategy from which no player would choose to deviate from their strategy even with knowledge of the other players' moves. Each player's strategy may be a *pure* strategy or a *mixed* (randomized) strategy. A mixed strategy is a probability distribution over a player's possible moves (those of non-zero probability are called the support of the mixed strategy). The pure strategy Nash equilibrium is found by determining, for each player, that player's strategy of best response to each of their opponents' strategies. The mixed strategy Nash equilibrium is found by determining which probability distribution of each player would cause their opponents to be indifferent between their own strategies - i.e., that none of the opponent strategies could do better than any other given the player's mixed strategy. The question of how complex it is to construct such an equilibrium has been dubbed "a most fundamental computational problem whose complexity is wide open" and "together with factoring, [...] the most important concrete open question on the boundary of P today" [46]. A Nash equilibrium is more formally described as follows:

Definition 1.2 (Nash equilibrium)**.** *Let $G = (\{f_S i\}, \{U_i\})$ be an n-player game and $p_1, \cdots, p_n$ a collection of probability distributions on the strategy sets. Distributions $p_1, \cdots, p_n$ are a Nash equilibrium if, for each player i, picking a strategy from S_i according to distribution p_i maximizes i's expected payoff when each player $\neq$ i picks a strategy according to the distribution p_j.*

A core problem in algorithmic game theory is the computation of a Nash equilibrium of a game. Denote *r*-Nash the Nash equilibrium problem on *r* players. When one analyzes the strategic structure of a game, especially from the viewpoint of a mechanism designer who tries to construct good rules for a game, finding a single equilibrium is far from satisfactory. More desirable equilibria may exist: in this case the game becomes more attractive, especially if one can coax the players into playing a desirable equilibrium. Also, less desirable equilibria may exist: in this case the game becomes less attractive.

Before we can make a definite judgment about the quality of the game, we would like to know the answers to questions such as: What is the game's most desirable equilibrium? Is there a unique equilibrium? If not, how many equilibria are there? Algorithms that tackle these questions would be useful both to players and to the mechanism designer.

Furthermore, algorithms that answer certain existence questions may pave the way to designing algorithms that construct a Nash equilibrium. For example, if we had an algorithm that told us whether there exists any equilibrium where a certain player plays a certain strategy, this could be useful in eliminating possibilities in the search for a Nash equilibrium. In this survey, we overview on problems of this nature.

A *fully mixed Nash equilibrium* is a Nash equilibrium in which all players assign a positive probability to each strategy of their strategy set. A *uniform* Nash equilibrium is a profile in which all players use a uniform probability distribution on their support. Other families or variants of Nash equilibria include:

A mixed Nash equilibrium of a game is *symmetric* if it is invariant under all automorphisms of the game. It is well known that every game has a symmetric mixed Nash equilibrium.

Pareto-optimal Nash Equilibria A distribution over mixed profiles is *Pareto-optimal* if there is no other distribution over outcomes such that every player has at least equal expected utility, and at least one player has strictly greater expected utility.

In an ϵ*-Nash equilibrium* each player can not gain more then ϵ times its current gain if moving to an alternative strategy. This is a relaxation of Nash equilibrium that aims to compensate for the inherent intractability of Nash equilibria.

Strong Nash Equilibria A *strong Nash equilibrium* [5] is a pure Nash equilibrium where no change of strategies of whatever coalition (i.e., group of players) can simultaneously increase the utility for all players in the coalition.

1.1.2 Correlated Equilibria

Despite the progress in the last years on polynomial time algorithms for computing Nash equilibria, it is not clear that such an algorithms will soon be found, or even that such an algorithm exists. This difficulty motivated researchers to consider weakening versions of the problem. *Correlated Equilibria* is a relaxed version of Nash equilibria. In particular, every Nash equilibrium is a correlated equilibrium and a correlated equilibrium of a game can be found in time polynomial in the standard description of the game. In fact, the set of all correlated equilibria can be described by a set of linear inequalities whose size is polynomial in the length of the game's description. Correlated Nash equilibria are more formally described as follows:

Definition 1.3. *Let $G = (\{f_S i\}, \{U_i\})$ be an n-player game. Let p be a probability distribution on $S_1, \cdots, S_n$. Distribution p is a correlated equilibrium if for each player i and each pair l, l' of strategies in S_i,*

$$\sum_{s:s_i=l} p(s)u_i(s) \geq \sum_{s:si=l} p(s)u_i(s')$$

, where s' is obtained from s by reassigning i's strategy to be l'.

One interpretation of a correlated equilibrium is as follows. A trusted authority picks a strategy profile s at random according to p, and "recommends" strategy s_i to player i. Each player i is assumed to know only its recommended strategy, and not those for other players. Player i can then compare the conditional expected payoffs of its strategies, assuming that the other players follow their recommendations (the conditioning is on the strategy recommended to i). The above inequality states that this conditional expectation should be maximized by the recommended strategy. If this holds for all players, then no player has a unilateral incentive to deviate from the trusted authority's recommendation. A popular concrete example of a correlated equilibrium is a traffic signal that recommends "red" (stop) or "green" (go) to oncoming drivers (see e.g. [43]).

2 Complexity Classes

We briefly describe the most important computational complexity classes that have been enriched with problems related to the computation of Nash equilibria and rational behavior of agents. For more details, we refer the reader to the classical textbook of Papadimitriou [44]. In all cases, a problem Π is hard for a complexity class C, under a given type of reduction if every problem in C reduces to Π using that reduction. A decision problem Π is complete for a class C (or, C-complete) if (i) it is in C and (ii) every problem in C reduces to it.

Class P The complexity class P is the set of problems that can be solved by a deterministic machine in polynomial time. This class corresponds to an intuitive idea of the problems which can be effectively solved in the worst cases.

Class $\mathcal{NP}$ The complexity class $\mathcal{NP}$ is the set of decision problems that can be solved by a non-deterministic machine in polynomial time. This class contains many problems that people would like to be able to solve effectively, including the Boolean satisfiability problem, the Hamiltonian path problem and the Vertex cover problem [22]. All the problems in this class have the property that their solutions can be checked effectively. For the problems that are complete in this class, it is unlikely to admit a polynomial algorithm for their solution. Obviously, $\mathsf{P} \subseteq \mathcal{NP}$. If the $\mathcal{NP}$ class is larger than P, then no easily scalable solution exists for these problems; whether this is the case remains to be an important open question in computational complexity theory.

A decision problem C is $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and if every other problem in $\mathcal{NP}$ is reducible to it [9]. "Reducible" here means that for every $\mathcal{NP}$ problem L, there is a polynomial time algorithm which transforms instances of L into instances of C, such that the two instances have the same truth values. As a consequence, if we had a polynomial time algorithm for C, we could solve all $\mathcal{NP}$ problems in polynomial time. For all known $\mathcal{NP}$-complete problems no efficient solution algorithm has been found and this is unlikely to happen. Many significant computer-science problems belong to this class–e.g., the traveling salesman problem (q.v.), satisfiability problems, and graph covering problems.

Class $\mathcal{PLS}$ The class $\mathcal{PLS}$, defined in [29], contains all *local search problems* with certain properties. A local search problem Π is given by its set of instances $\mathcal{I}_\Pi$. For

every instance $I \in \mathcal{I}_\Pi$, we are given a finite set of feasible solutions $\mathcal{F}(I)$, an objective function$c : \mathcal{F}(I) \in\rightarrow \mathbb{Z}$, and for every feasible solution $S \in \mathcal{F}(I)$, a *neighborhood* $N(S, I) \subseteq F(I)$. Given an instance I of a local search problem, we seek for a *locally optimal solution* S^*, i. e., a solution which does not have a strictly better neighbor. A neighbor S' of a solution S is strictly better if the objective value $c(S')$ is larger/smaller in the case of a maximization/ minimization problem. A local search problem Π belongs to $\mathcal{PLS}$ if the following polynomial time algorithms exist:

1. an algorithm A which computes for every instance I of Π an initial feasible solution $S_0 \in \mathcal{F}(I)$,

2. an algorithm B which computes for every instance I of Π and every feasible solution $S \in \mathcal{F}(I)$ the objective value $c(S)$,

3. an algorithm C which determines for every instance I of Π and every feasible solution $S \in \mathcal{F}(I)$ whether S is locally optimal or not and finds a better solution in the neighborhood of S in the latter case.

Given an instance I of a local search problem Π, we denote by $TG(I)$ the *transition graph* that contains a node $v(S)$ for every feasible solution $S \in \mathcal{F}(I)$ and a directed edge from a node $v(S_1)$ to a node $v(S_2)$ if S_2 is in the neighborhood of S_1 and if the objective value $c(S_2)$ is strictly better than the objective value $c(S_1)$.

$\mathcal{PLS}$ captures an important computational paradigm, that of local optimization. Local optimization gives rise to some of the most successful heuristics for a host of hard combinatorial optimization problems.

$\mathcal{PLS}$-completeness requires a rather specialized and involved kind of reduction, called $\mathcal{PLS}$-reduction. A $\mathcal{PLS}$-reduction from problem A to B is a polynomial mapping from instances of A to instances of B, plus a polynomial mapping recovering from any local optimum of B a local optimum of A.

In [29] two problems were shown to be $\mathcal{PLS}$-complete and thus as hard as any problem in $\mathcal{PLS}$; they were a "generic" problem called FLIP, and the problem of finding a local optimum in the Kernighan- Lin heuristic for the graph partitioning problem [32].

[49] showed that finding a local optimum under the Lin-Kernighan heuristic for the traveling salesman problem is $\mathcal{PLS}$-complete. Also, they showed that finding stable configurations in neural networks in the Hopfield model is $\mathcal{PLS}$-complete.

Class $\mathcal{PPAD}$ A *search problem* $\mathcal{S}$ is a set of inputs $I_\mathcal{S} \subseteq \Sigma^*$ such that for each $x \in I_\mathcal{S}$ there is an associated set of solutions $\mathcal{S}_x \in \Sigma^{|x|^k}$ for some integer k, such that for each $x \in I_\mathcal{S}$ and $y \in \Sigma^{|x|^k}$ whether $y \in \mathcal{S}_x$ is decidable in polynomial time (notice that this is precisely $\mathcal{NP}$ with an added emphasis on finding a witness).

For example, r-Nash (finding a Nash equilibrium for a game with r players) is the search problem $\mathcal{S}$ in which each $x \in I_\mathcal{S}$ is an r-player game in normal form together with a binary integer A (the accuracy specification), and $\mathcal{S}_x$ is the set of $\frac{1}{A}$-Nash equilibria of the game.

A search problem is *total* if $\mathcal{S}_x \neq= ;$ for all $x \in I_\mathcal{S}$. For example, Nash's 1951 theorem [42] implies that r-Nash is total. Similarly, d-graphical Nash (finding a Nash equilibrium for a graphical game with a graph of degree at most d) The set of all total search problems is denoted TFNP.

The interesting TFNP functions come from combinatorial theorems that guarantee solutions. One such theorem, called the Polynomial Parity Argument (PPA), is given a finite graph consisting of lines and cycles (every node has degree at most 2), there is an even number of endpoints.

The class $\mathcal{PPAD}$ is defined using directed graphs based on PPA. Formally, $\mathcal{PPAD}$ is defined by its complete problem (from an earlier post), called End of the Line): Given an exponential-size directed graph with every node having in-degree and out-degree at most one described by a polynomial-time computable function $f(v)$ that outputs the predecessor and successor of v, and a vertex s with a successor but no predecessors, find a $s \neq t$ that either has no successors or predecessors.

Class $\mathcal{NEXP}$ The complexity class $\mathcal{NEXP}$ (also called $\mathcal{NE}$) is the s the set of decision problems that can be solved by a non-deterministic Turing machine using time $O(2^{p(n)})$.

An important set of $\mathcal{NEXP}$-complete problems relates to succinct circuits. Succinct circuits are simple machines used to describe graphs in exponentially less space. They accept two vertex numbers as input and output whether there is an edge between them. If solving a problem on a graph in a natural representation, such as an adjacency matrix, is $\mathcal{NP}$-complete, then solving the same problem on a succinct circuit representation is $\mathcal{NEXP}$-complete, because the input is exponentially smaller.1 As one simple example, finding a Hamiltonian path for a graph thus encoded is $\mathcal{NEXP}$-complete.

Polynomial Hierarchy The *polynomial hierarchy* Σ_k^p is a hierarchy of complexity classes that generalize the classes P, $\mathcal{NP}$ and $co-\mathcal{NP}$ to oracle machines. (An *oracle machine* is an abstract machine used to study decision problems. It can be visualized as a Turing machine with a black box, called an oracle, which is able to decide certain decision problems in a single step.)

An equivalent definition in terms of alternating Turing machines defines Σ_k^p as the set of decision problems solvable in polynomial time on an alternating Turing machine with k alternations starting in an existential state. It holds that $\Sigma_0^P = \mathsf{P}$, $\Sigma_1^P = \mathcal{NP}$. The union of all classes in the polynomial hierarchy is the complexity class $\mathcal{PH}$.

An example of a natural problem in Σ_2^p is circuit minimization: given a number k and a circuit A computing a Boolean function f, determine if there is a circuit with at most k gates that computes the same function f.

Class #P #P the set of counting problems associated with the decision problems in the set $\mathcal{NP}$. Unlike most well-known complexity classes, it is not a class of decision problems but a class of function problems.

More formally, a problem is in #P if there is a non-deterministic, polynomial-time Turing machine that, for each instance I of the problem, has a number of accepting computations that is exactly equal to the number of distinct solutions for instance I.

Clearly, a problem must be at least as hard as the corresponding $\mathcal{NP}$ problem. If it's easy to count answers, then it must be easy to tell whether there are any answers. Just count them, and see if the count is greater than zero. Therefore, the #P problem corresponding to any $\mathcal{NP}$-Complete problem, must be $\mathcal{NP}$-Hard.

Surprisingly, some #P problems that are believed to be difficult correspond to easy P problems.

3 Games

3.1 Succinct Games

In this section, we provide simple but sufficient definitions of some of the most important games, that have been investigated the last decades, and we will present next various results. *Multiplayer games* not only represent more realistic scenaria but also are also the most compelling specimens in the study of computing Nash equilibria. But, to be of algorithmic interest, they must be *represented succinctly* or *compactly*. Succinct representation is required since otherwise a typical (multiplayer) game would need an exponential size of bits in order to be described. Some well known games that admit a succinct representation include:

- *Symmetric games* [48], where all players are identical and indistinguishable,
- *Graphical games* [33], where the players are the vertices of a graph, and the payoff for each player only depends on its own strategy and those of its neighbours;
- *Congestion games* [50], where the payoff of each player only depends on its strategy and those choosing the same strategy as him.
- *Games on graphs* [1, 38], where the strategy set of all players is given by a graph.

In the next sections, we briefly describe each one of the three families of succinct games mentioned above. More formally a succinct game is defined as follows:

Definition 3.1. *A* succinct game $\mathcal{G} = (\mathcal{I}, \mathcal{T}, \mathcal{U})$ *is defined, like all computational problems, in terms of a set of inputs* $I \in \mathsf{P}$*, and two polynomial algorithms* T *and* U*. For each* $z \in I$*,* $T(x)$ *returns a* type*, that is, an integer* $n \geq 2$ *(the number of players) and an* n*-tuple of integers* $(t_1, \cdots, t_n)$*, each at least* 2 *(the cardinalities of the strategy sets). If* n *and the* t_i*'s are polynomially bounded in* $|x|$*, the game is said to be of* polynomial type*. Given any* n*-tuple of positive integers* $s = (s_1, \cdots, s_n)$*, with* $s_i \leq t_i$ *for all* i*, and* $i \leq n$*,* $U(z, i, s)$ *returns an integer standing for the utility* $u^i(s)$*. The resulting game is denoted* $\mathcal{G}(\ddagger)$*.*

[2] considered a way of describing the set of actions so that they are not given explicitly and directly, by listing all their actions, but succinctly and implicitly. [2] are interested in descriptions whose length does not depend dramatically on the number of the actions, but depends on the length of the actions. Such descriptions are exponentially more succinct than the sets they describe. The following definitions captures this idea.

- **Strategic Games in Implicit Form**. A game is a tuple $\Gamma = \langle 1^n, 1^m, M, 1^t \rangle$. This game has n players. For each player i, their set of actions is $A_i = \sum^m$ and $\langle M, 1^t \rangle$ is the description of the pay-off functions.

  The second family of games is defined by considering that the set of actions of each player is given explicitly.

- **Strategic Games in General Form**. A game $\Gamma = \langle 1^n, A_1, \cdots, A_n, M, 1^t \rangle$, has n players, for each player i, their set of actions A_i is given by listing all its elements. The description of their pay-off functions is given by $\langle M, 1^t \rangle$.

Finally, we consider a less succinct description of games. This is the usual description adopted in basic books giving us a complete description in form of a bimatrix or trimatrix (set of bimatrices).

- **Strategic Games in Explicit Form**. A game is a tuple $\Gamma = \langle 1^n, A_1, \cdots, A_m, T \rangle$. It has n players, and for each player i, their set of actions A_i is given explicitly. T is a table with an entry for each strategy profile a and a player i. In this case $u_i(a) = T(a, i)$.

3.2 Symmetric Games

In a *symmetric* game [48], all players are identical and indistinguishable. They have the same strategy sets, their utility functions are the same function of their own strategy and the other players' actions, and this function is symmetric in the other players' actions. This function thus depends only on the number of players choosing each strategy, and on the player's strategy.

Symmetric games have been widely studied since the dawn of game theory. For example, Nash [42] proved that any symmetric game must have a *symmetric* equilibrium;-an equilibrium in which all players play the same strategy. Also, many of game theory's most famous examples are symmetric (e.g., the Prisoner's Dilemma, Chicken, coordination games). Symmetric games can only be easier to design algorithms for than general non-cooperative games.

However, in a more subtle sense, symmetric games can be more difficult than general games. In particular, the utility functions of an n-player k-strategy symmetric game can be specified with $k \cdot n + k \binom{k}{-} 1$ numbers. For $k = O(n)$ strategies, this is much less than the nk^n numbers required to a describe a general game in which all n players have k strategies available.

Many of the examples mentioned above, such as the n-player Prisoner's Dilemma satisfy $k = O(n)$ or even $k = O(1)$. If such a symmetric game is represented succinctly (arguably the most natural way), it is more difficult for an algorithm to run in time polynomial in the input size. An important question is then: what positive algorithmic results for general games (e.g., for correlated equilibria) carry over to symmetric games that are represented compactly?

A symmetric game can be more formally defined as follows:

Definition 3.2. *[48] A game is symmetric if $S_1, \cdots, S_n$, $u_i(s)$ depends only on s_i and the other players' strategies (but not on i), and $u_i(s)$ is a symmetric function of the variables $s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_n$. In other words, the payoff to a player depends only on its strategy and on the number of players choosing each of the different strategies in S.*

A symmetric game can be specified by giving, for each ordered partition of the number of players, the payoffs of each player $k \cdot n + k \binom{k}{-} 1$, where $k = |S|$. This expression is $\Theta(n^{k-1})$ when $k = O(1)$, polynomial in n^k when $k = O(\log n / \log \log n)$, and is super-polynomially smaller than k^n unless $k = \Omega(n^{1+\epsilon})$. A symmetric Nash equilibrium is defined as follows:

Definition 3.3. *Let G be a symmetric n-player game. A Nash equilibrium $p_1, \cdots, p_n$ is symmetric if $p_1 = \cdots = p_n$. Nash [42] proved that every symmetric game admits a symmetric Nash equilibrium (a symmetric game may, of course, have other Nash equilibria).*

3.3 Graphical Games

Graphical Games were first proposed by Kearns, Littman, and Singh [33]. In a graphical game, the players are the vertices of a graph, and the payoff of each player only depends on its strategy and those of its neighbors. More specifically, an n-player game is given by an undirected graph G on n vertices and a set of n matrices specifying the payoffs to each player under any possible combination of joint actions. The interpretation is that the payoff to player i is determined entirely by the actions of player i and his neighbors in the graph, and thus the payoff matrix for player i is indexed only by these players. $N(i) \subseteq \{1, \cdots, n\}$ denotes the *neighborhood* of player i in G Ůthat is, those vertices j such that the edge (i, j) appears in G. By convention $N(i)$ always includes i itself as well. If $\vec{a}$ is a joint action, we use $\vec{a}^i$ to denote the induced vector of actions just on the players in $N(i)$. Formally, we define:

Definition 3.4. *[31] A graphical game is a pair* $(\mathcal{G}, \mathsf{M})$*, where* $\mathcal{G}$ *is an undirected graph over the vertices* $\{1; \cdots, n\}$*, and* M *is a set of n local game matrices. For any joint action* $\vec{a}$*, the local game matrix* $M_i \in \mathsf{M}$ *specifies the payoff* $M_i(\vec{a}^i)$ *for player i, which depends only on the actions taken by the players in* $N(i)$*.*

Graphical games are a potentially more compact way of representing games than standard normal form. In particular, rather than requiring size exponential in the number of players n, a graphical game requires size exponential only in the size d of the largest local neighborhood. Thus if $d << n$, the graphical representation is exponentially smaller than the normal form. Note that we can represent any normal form game as a graphical game by letting G be the complete graph, but the representation is most useful when a considerably sparser graph can be found.

We thus view the global n-player game as being composed of interacting local games, each involving (perhaps many) fewer players. Each playerŠs action may have global impact, but it occurs through the propagation of local influences.

There are many common settings in which such graphical models may naturally and succinctly capture the underlying game structure. The graph topology might model the physical distribution and interactions of agents: each salesperson is viewed as being involved in a local competition (game) with the salespeople in geographically neighboring regions. The graph may be used to represent organizational structure: low-level employees are engaged in a game with their immediate supervisors, who in turn are engaged in a game involving their direct reports and their own managers, and so on up to the CEO. The graph may coincide with the topology of a computer network, with each machine negotiating with its neighbors.

The notions of neighborhood of a player can be also defined for a general game. So, for any game we can define:

Bounded Neighborhood Game or k-graphical game Let $k > 0$ be a fixed constant. A strategic game with associated neighborhood function $N()$ has k-bounded neighborhood if, for each player i, $|Neigh(i)| \leq k$.

Note that for any strategic game, we can define the notion of bounded neighborhood. When we refer to a graphical game, we mean a game that is actually (compactly) described in a using a graph and related notions.

3.3.1 Highly Regular Games

When the graph of interest is highly regular, perhaps the $n \times n$ grid (torus), and all players are locally identical, the representation of such a game is extremely succinct: Just the game played at each locus, and the size of the grid at each dimension. The total input size becomes $O(s^5 + \log n)$, where n is the number of players and s is the number of strategies of each player. These games, are called *highly regular graph* games.

A torus game is *fully symmetric* if it has the additional property that the utility function u is symmetric with respect to the 2d neighbors of each node. Our negative results will hold for this special case, while our positive results will apply to all torus games.

3.4 Congestion Games

Congestion games are an abstraction of network routing games and were first defined by Rosenthal [50]. In a congestion game, there is a ground set of elements, and players choose a strategy from a prescribed collection of subsets of the ground set. The cost of an element is a function of the number players that select a strategy that contains it, but this cost is independent of the identities of these players. The cost (negative payoff) to a player is then the sum of the costs of the elements in its strategy.

We shall consider games in which the players utility functions u_i's are given implicitly in terms of efficient algorithms computing the utilities based on the input and the state. For example, in a congestion game the input is a set of n players, a finite set E of *resources*, and the action sets are $S_i \subseteq 2^E$; we are also given the *delay function* d mapping $E \times \{1, cdots, n\}$ to the integers. $d_e(j)$ is nondecreasing in j. The payoffs are computed as follows. Let $s = (s_1, \cdots, s_n)$ be a state, and let $f_s(e) = |\{i : e \in s_i\}$. Then $c_i(s) = -u_i(s) = \sum_{e \in s_i} d_e(f_s(e))$. Intuitively, each player chooses a set of resources (from among the sets available to her), and to compute the cost incurred by i (the negative of her payoff) we add the delay of each resource used by i, where the delay of a resource e depends on the congestion $f_s(e)$, the total number of players using e.

In a *network congestion game* the families of sets S_i are presented implicitly as paths in a network. We are given a network (V, E), two nodes a_i, $b_i \in V$ for each player i and again a delay function with the edges playing the role of the resources. The subset of E available as actions to the player i is the set of all paths from $a_i$to b_i. We shall assume the network is directed.

A function $P : S \to \mathbb{Z}$ is called *exact potential* for a game if, for any $s, s' \in S$ which differ only in the i-th component,

$$P(s') - P(s) = u_i(s') - u_i(s).$$

A game is called *potential* if it admits a potential function. In his seminal work, Rosenthal showed that potential and congestion games are related:

Theorem 3.5 ([50]). *Every congestion game is a potential game.*

The same work shows that every potential game admits a pure Nash equilibrium. It follows that:

Theorem 3.6 ([50]). *Every congestion game has a pure Nash equilibrium.*

A network congestion (or more generally, a potential) game is *symmetric* if all players have the same endpoints a and b (and thus they all have the same set of paths/strategies).

By the proof of Rosenthal's Theorem above, finding a pure Nash equilibrium for a congestion game is in $\mathcal{PLS}$, as it is equivalent to finding a local optimum of ϕ, where the feasible solutions are all states. Notice that this does not imply a polynomial algorithm, since improvements of ϕ can be small and exponentially many.

KP Model Koutsoupias and Papadimitriou considered a very simple weighted network congestion game, now known as *KP-model*. Here, the network consists of a single source and a single destination (*single-commodity* network) which are connected by parallel links. The *load* on a link is the total weight of players assigned to this link. Associated with each link is a *capacity* representing the rate at which the link processes load. Each of the players selfishly routes from the source to the destination by using a probability distribution over the links. The private objective function of a player is defined as its expected latency. In the KP-model, the *Social Cost* is defined as the expected maximum latency on a link, where the expectation is taken over all random choices of the players.

More formally, in KP model, each of the n players is allowed to use exactly one of m resources (called also *links*), that is, $S_i = [m]$ for all $i \in [n]$. The players are called *identical* if all weights (user demands) are equal, otherwise *arbitrary* (the corresponding game is called *weighted congestion game*). Associated with each link $j \in [m]$ is a *capacity* c_j representing the rate at which link j processes *load*. Clearly, the latency on link j is $f_j(l_j) = \frac{l_j}{c_j}$, showing that the latency functions are linear. If $c_1, \cdots, c_m$ are equal, then the resources are *identical*, otherwise *related*.

In (weighted) *network congestion games*, the strategy sets of the players correspond to paths in a network. We call a congestion game a *matroid* congestion game if the combinatorial structures of the players' strategy spaces are matroids.

Threshold games *Threshold games* are a special class of congestion games in which the set of resources $\mathbb{R}$ is divided into two disjoint subsets $\mathbb{R}_{\text{in}}$ and $\mathbb{R}_{\text{out}}$. The set $\mathbb{R}_{\text{out}}$ contains a resource r_i for every player $i \in \mathcal{N}$. This resource has a fixed delay T_i called the *threshold* of player i. Each player i has only two strategies, namely a strategy $S_i^{out} = \{r_i\}$ with $r_i \in R_{out}$, and a strategy $S_i^{in} \in\subseteq \mathbb{R}_{\text{in}}$. The preferences of player i can be described in a simple and intuitive way: Player i prefers to choose strategy S_i^{in} against strategy S_i^{out} if the delay of S_i^{out} is smaller than the threshold T_i. *Quadratic threshold games* are a quite restrictive subclass of threshold games. In this variant, the set $\mathbb{R}_{\text{out}}$ contains exactly one resource r_{ij} for every unordered pair of players $\{i, j\} \subseteq \mathcal{N}$.

Overlay Network Design Game In an *overlay network design game* we are given an undirected graph $G = (V, E)$ with a delay function $d_e : \mathbb{N} \to \mathbb{N}$ for every edge $e \in E$ and a fixed routing path between any pair of nodes.

Atomic Congestion Games among Coalitions Most of known noncooperative strategic games are motivated from network traffic and congestion problems. But real life examples justify the necessity for the consideration of selfish coalitions, since in most cases there is some sort of hierarchy that needs to be taken into account. For example, in a communication network, the service providers are interested in minimizing their own

cost for assuring a promised quality of service (eg, bandwidth) to their users, but they are not actually interested in utilizing their usersŠ individual delays, so long as the maximum delay is small. In fact, in some cases they have to sacrifice the utilization of some of their users dictatorially, for the sake of their own (still private and selfish wrt other coalitions) objective. In this scenario, each service provider can be seen as a static coalition of users that tries to minimize the maximum cost that any of its users has to pay. Alternatively, we may actually have altruistic (wrt their participating users) coalitions that try to minimize the cumulative cost that all their users have to pay, by performing joint decisions for all their members.

All these scenarios are captured by *(static) coalitional congestion games*, in which the selfish players are actually the coalitions that may handle more than one users at will. Coalitions of players in atomic congestion games implies that each coalition selfishly governs a unique subset of atomic players, whose traffic demands have to be routed via single paths (ie, unsplittably). Nevertheless, a coalition is allowed to choose different routes for different players that it handles. More formally, such a game can be defined, based on the KP model, as follows:

Definition 3.7 (Coalitional KP model)**.** *Consider a collection* $[m]^1$ *of identical parallel links and a collection* $[n]$ *of tasks. Each task must be uniquely allocated to any of the m available links. Each task* $j \in [n]$ *has an integer* demand $w_j \in \mathcal{N}_+$ *(eg, the number of elementary operations for the execution of task j). Let* $\widetilde{W} = \{w_j\}_{j\in[n]}$ *be the multiset of the tasks' demands. A set of* $k \geq 1$ *(static) coalitions* $C_1, \cdots, C_k$ *is a partition of* $\widetilde{W}$ *into k nonempty multisets. Hence: (i) the union (as multisets) of these coalitions is exactly* $\widetilde{W}$*, (ii)* $C_j \neq \emptyset$*,* $\forall j \in [k]$*, and (iii)* $C_i \cap C_j = \emptyset$ *,* $\forall i, j \in [k] : \ i \neq j$*. For* $j \in [k]$ *let* $C_j = \{w_j^1, \cdots, w_j^{n_j}\}$*, so that* $\sum_{j=1}^{k} n_j = n$*. Denote by* $W_j = \sum_{i=1}^{n_j} w_j^i$ *the cumulative demand required by coalition* C_j *, while* $W_{tot} = \sum_{j\in[k]} W_j$ *is the overall demand required by the system. The* expected load *on link* $l \in [m]$ *is the expectation of the load on link l according to a given profile.*

Definition 3.8 (Network congestion game with coalitions)**.** *Let* $G = (V, E)$ *be a directed network with a non-decreasing delay function* $d_e(x)$ *for each* $e \in E$*. Consider also a multiset of users of* identical traffic demands*, willing to be routed between unique source-destination pairs of nodes in the network. A* network congestion game with coalitions *consists of a set of players giving the set of coalitions* $\{C_1, \cdots, C_k\}$*. Every coalition* C_j *consists of* n_j *users routing their traffic from* s_j *to* t_j*.*

There are (at least) two natural notions of selfish cost of coalition C_j *in a profile* σ*. The first is the* maximum delay *over all paths used by* C_j *and the second is the total delay of coalition. Both maximum delay and total delay generalize the notion selfish cost used in congestion games in a natural way.*

3.5 Market Sharing Games

Market Sharing games have been introduced by Goemans et al. [23] to model non-cooperative content distribution in wireless networks. An instance of a market sharing game consists of a set of n players $\mathcal{N}$, a set $\mathsf{M} = \{1, \cdots, l\}$ of markets, and a bipartite graph $G = (\mathcal{N} \cup \mathsf{M}, E)$. An edge between player i and market r indicates that player i is interested in market m. Furthermore, for each market m, costs c_m and a so-called *query rate* $q_m \in \mathbb{N}$ are given, and, for each player i, a budget B_i is specified. The query

rate q_m determines the payoff of market m which is equally distributed among the players who have allocated that market, i. e., the payoff function of market m is given by $p_m(n_m) = q_m/n_m$. We define the *delay of a market* to be equal to the negative payoff of the market.

3.6 Games on Graphs

In the *games on graphs*, the strategy sets of the players is specified by a graph $G(V, E)$; each player's strategy set is obtained by the entities involved in the related graph: the set of vertices, the set of edges, etc. The resulting game instance is highly depended on the selection of the graph. A powerful property of such games is that for their study, one could explore, besides tools of Game Theory, tools of the rich field of *Graph Theory*. Note that such games are succinct games, since the strategy set of all players is given by a polynomial size structure; a graph G.

Indeed, two independent research teams, one consisting of Aspnes *et al.* [34, 1] and another consisting of Mavronicolas *et al.* [39, 38], initiated recently the introduction of strategic games on graphs (and the study of their associated Nash equilibria) as a means of studying *security problems in networks with selfish entities*. The non-trivial results achieved by these two teams exhibit a novel interaction of ideas, arguments and techniques from these two seemingly diverse fields, namely *Game Theory* and *Graph Theory*. This research line invites a simultaneously game-theoretic and graph-theoretic analysis of network security problems, where not only threats seek to maximize their caused damage to the network, but also the network seeks to protect itself as much as possible. In the following we present the related games.

3.6.1 Virus Inoculation Games

Definition 3.9. *[34, 1] The network topology is represented by an undirected graph $G = (V, E)$, where V is a set of network hosts and $E = V \times V$ is a set of (bidirectional) communication links. The basic model for installing anti-virus software is a one-round game:*

- **Players***: The game has n players corresponding to nodes of the graph. Initially, all nodes are insecure and vulnerable to infection.*

- **Strategies***: We denote the strategy of i by a_i. Each node i has two possible actions: do nothing and risk being infected or inoculate itself by installing anti-virus software. Node i's strategy a_i is the probability that it inoculates itself. Nodes' choices can be summarized in a strategy profile $\vec{a} \in [0, 1]^n$. If a_i is 0 or 1, we say that node i adopts a pure strategy; otherwise, its strategy is mixed. We call nodes that install anti-virus software secure and denote the set of such nodes by $I_{\vec{a}}$. We associate an attack graph $G_{\vec{a}} = G - I_{\vec{a}}$ with $\vec{a}$.*

- **Attack model***: After the nodes made their choices, the adversary picks some node uniformly at random as a starting point for infection. Infection then propagates through the network graph. Node i gets infected if it has no anti-virus software installed and if any of its neighbors become infected.*

- **Individual costs**: *Suppose it costs C to install anti-virus software. If a node is infected, it suffers a loss equal to L. For simplicity, we assume that both C and L are known and are the same for all nodes. The cost of a mixed strategy* $\vec{a} \in [0,1]^n$ *to node i is*

$$cost_i(\vec{a}) = a_i C = (1 - a_i) \cdot L \cdot p_i(\vec{a}).$$

The virus inoculation game *obtained is given by the triple* (G, C, L).

3.6.2 Attackers-Defender Games

The work of Mavronicolas *et al.* [38, 39] considers a security problem on a distributed network modeled as a multi-player non-cooperative game with *attackers* (e.g., viruses) and a *defender* (e.g., a security software) entities. More specifically, there are two classes of confronting randomized players on a graph: ν *attackers*, each choosing vertices and wishing to minimize the probability of being caught, and a single *defender*, who chooses edges and gains the expected number of attackers it catches. More formally the game is defined as follows:

Definition 3.10. *[38] Associated with G is a* strategic game $\langle \mathcal{N}, \{S_i\}_{i \in \mathcal{N}}, \{\mathsf{IP}\}_{i \in \mathcal{N}} \rangle$ *on G defined as follows:*

- *The set of* players *is* $\mathcal{N} = \mathcal{N}_{vp} \cup \mathcal{N}_{ep}$*, where:*
  - $\mathcal{N}_{vp}$ *is a finite set of* ν vertex *players* vp_i*, called* attackers*,* $1 \leq i \leq \nu$*;*
  - $\mathcal{N}_{ep}$ *is a singleton set of an* edge *player ep, called* defender.
- *The strategy sets of the players are as follows:*
  - *The* strategy set S_i *of vertex player* vp_i *is* V.
  - *The* strategy set S_{ep} *of the edge player ep is* E.

  So, the strategy set $\mathcal{S}$ *of the game is* $\mathcal{S} = \left(\underset{i \in \mathcal{N}_{vp}}{\times} S_i \right) \times S_{ep} = V^\nu \times E$.
- *Fix an arbitrary* strategy profile $\vec{\mathbf{s}} = \langle s_1, \ldots, s_\nu, s_{ep} \rangle \in \mathcal{S}$*, also called a* profile.
  - *The* Individual Profit *of vertex player* vp_i *is a function* $\mathsf{IP}_{\vec{\mathbf{s}}}(i) : \mathcal{S} \to \{0, 1\}$ *such that* $\mathsf{IP}_{\vec{\mathbf{s}}}(i) = \begin{cases} 0, & s_i \in s_{ep} \\ 1, & s_i \notin s_{ep} \end{cases}$ *; intuitively, the vertex player* vp_i *receives* 1 *if it is not caught by the edge player, and* 0 *otherwise.*
  - *The* Individual Profit *of the edge player ep is a function* $\mathsf{IP}_{\vec{\mathbf{s}}}(ep) : \mathcal{S} \to \mathbb{N}$ *such that* $\mathsf{IP}_{\vec{\mathbf{s}}}(ep) = |\{i : s_i \in s_{ep}\}|$*; intuitively, the edge player ep receives the number of vertex players it catches.*

3.7 Various Games

A 2-player game is called *zero-sum* if in each possible strategy profile of the game, the sum of players utilities equals to zero.

In a *polymatrix* game [26] each player plays a 2-player game with each other player, and his/her choices are the same in each of these games; the utilities are then added.

In a *hypergraphical* game several subsets of the players are involved in separate games with utilities added. If the hypergraph is a clique graph, we have a bimatrix game; if every vertex has nonzero utility in only one hyperedge, and these nonzero hyperedges have a certain symmetry property, we have a graphical game. Hypergraphical games generalizes both graphical and polymatrix games.

4 Class P

Here, we overview on problems for which their their computational complexity have been proven to be in the class P.

[P0]. **Instance:** A zero-sum, 2-player game.
Output: A Nash equilibrium.

Comment: Almost sixty years ago Morgenstern and von Neumann [40] initiated the study of game theory with their applications to Economic behavior. A particularly interesting mathematical result is their proof of the existence of equilibrium in the 2-player zero-sum game. It exploits duality properties of polytopes, which also lead to Dantzig's linear programming method [11] for optimization problems. Since linear programming is polynomial time solvable, by Khachian's ellipsoid algorithm [30], their result implies that computing a Nash equilibrium for a zero-sum, 2-players game is polynomial time solvable.

[P1]. **Instance:** A strategic game in general form, with k-players, for any $k \geq 2$.
Question: Does there exists a pure Nash equilibrium?

Comment: [2] showed that for strategic games in general (explicit) form, with k-players, the problem of deciding whether the game admits a pure Nash equilibrium is P-complete. Furthermore, they showed that the same problem for strategic games in explicit form is in $\mathcal{AC}'$. (The class $\mathcal{AC}'$ contains all decision problems solvable by circuits of polynomial size and constant depth.)

4.1 Symmetric Games

[P2]. **Instance:** An n-player, k-strategy symmetric game.
Output: A symmetric Nash equilibrium.

Comment: [48] showed that the problem can be solved in time polynomial in the size of its standard representation ($\Omega(k^n)$ numbers) and in the number of bits of precision desired if $k = O(n)$. If the game is represented compactly (using $\Omega(poly(n^k)$ numbers), then their

algorithm runs in polynomial time provided $k = O(\log n = \log \log n)$.

4.2 Compact Games

[P3]. INSTANCE: A tree graphical game $(\mathcal{G}, \mathsf{M})$.
OUTPUT: A correlated equilibrium.

COMMENT: [31] established a powerful relationship between the graphical structure of a multiplayer game and a certain Markov network representing distributions over joint actions. The first showed that this Markov network succinctly represents all correlated equilibria of the graphical game up to expected payoff equivalence.

Then they proceeded showing the polynomial time computation for a tree graphical game. For such a game, the traditional linear programming methods that use the standard *non-compact* representation of the game, do not succeed since they need exponential time. Thus, more a concise way to express correlated equilibrium and distributional constraints is needed; ideally linearly in the size of the graphical game representation. [31] showed this is indeed possible for tree graphical games. The basic idea is to express both the correlated equilibrium and distributional constraints entirely in terms of the local marginals, rather than the global probabilities of joint actions.

A more general result on succinctly represented games, such that graphical, congestion and symmetric games was the following:

[P4]. INSTANCE: A succinctly represented game.
OUTPUT: A correlated equilibrium.

COMMENT: [48] presents a general framework for optimizing over correlated equilibria of a game in time polynomial in the size of a succinct representation. Their approach using linear programming techniques. The framework applies in certain congestion games and graphical games:

- For a n-player, k-strategy symmetric game they showed that the problem can be solved in time polynomial in the size of its succinct representation.
- For graphical game $(\mathcal{G}, \mathsf{M})$, where G is a graph with bounded treewidth [51], they showed that the problem can be solved in polynomial time to its natural succinct representation.

In particular, [48] defined a class of succinct games encompassing all of the above except polymatrix and hypergraphical games: For each player the strategy profiles of all opponents are divided into equivalence classes, in which the playerŠs utility depends only on his/her own choice. The linear program (see (CE) below) describing correlated equilibria involves an exponential number of variables; such programs are usually solved by Şcolumn generationŤ variants of simplex. They developed an ellipsoid-based version of this technique taking advantage of the special structure of CE), whereby polynomial

solution is implied by the existence of a polynomial oracle for a particular simple problem involving strategy profiles and equivalence classes.

In [48] the computed correlated equilibrium is succinct in that it has polynomial support. [47] addressed another possible form of succinctness is the polynomial mixture of products, or *pmp*'s: A distribution over strategy space is given as the convex combination of polynomially many product (independent, Nash equilibrium-like) distributions; in other words, an n-dimensional tensor with polynomially small (positive) rank. Such a correlated equilibrium has an attractive implementation (sampling algorithm): First sample the products, and then sample the strategies of each player according to the resulting product distribution. The correlated equilibria computed by their algorithm are pmp's.

[47] Papadimitriou presented an algorithm that works on the dual (D) of (CE), which has polynomially many variables and exponentially many constraints. The existence proof implies (D) is infeasible; despite this fact, they run the ellipsoid algorithm on it, using Markov chain computations at each step to find a violated convex combination of the constraints of (D). At the conclusion of the algorithm, a a polynomial number of such combinations (the cuts of the ellipsoid algorithm) that are themselves infeasible is obtained, call those (D'). Solving the dual of this new linear program, called (P'), gives us the required correlated equilibrium as a pmp.

Even though (P') has polynomial dimensions, its constraint matrix must be computed as the product of two exponentially large matrices (the constraints of (P) times the matrix whose columns are the computed pmp's). The computation implicit in the matrix multiplication formula for each element of the constraint matrix of (P') suggests an expectation calculation; for each class of succinct games a problem dependent trick to carry this out is needed. Fortunately, in all cases outlined above, this turns out to always be doable in polynomial time, by utilizing one or two of three techniques: Linearity of expectation (polymatrix games, hypergraphical games), dynamic programming (congestion, scheduling, and local effect games), or implementing the definition of expectation on polynomially small domains (graphical and hypergraphical games).

Given a succinct game G, a *polynomial correlated equilibrium scheme* [47] consists of two polynomial time algorithms A and R. A on input $z \in$ produces (deterministically or not) an output y. R is a randomized algorithm which, on input z and y, will select an element s of the strategy space of $G(x)$ with probability corresponding to a correlated equilibrium of $G(z)$. [47] showed that for the following classes of succinct games of polynomial type have a polynomial correlated scheme:

- Polymatrix games;
- Graphical games;
- Hypergraphical games;
- Congestion games;
- Local effect games;
- Scheduling games;
- Facility location games;
- Symmetric games.

[P5]. **INSTANCE:** A game with a bounded hypertreewidth (a generalization of treewidth) and a small neighborhood or a graphical game with a bounded hypertreewidth.
OUTPUT: A pure Nash equilibrium.

COMMENT: [20] showed that the problem is feasible in polynomial time for all classes of games. Observe that each of the two joint restrictions, small neighborhood and bounded hypertree width, is weaker than the restrictions of bounded neighborhood and acyclicity, respectively, of which each by itself does not guarantee tractability. Thus, in order to obtain tractability, instead of strengthening a single restriction, they combined two weaker restrictions.

In order to prove the tractability result, they established a relationship between strategic games and the well-known finite domain constraint satisfaction problem (CSP), much studied in the AI and OR literature. They showed that each (general, not necessarily graphical) strategic game $\mathcal{G}$ can be translated into a CSP instance having the same hypertree width as $\mathcal{G}$, and whose feasible solutions exactly correspond to the Nash equilibria of the game. Then, they proved that $\mathcal{G}$ is equivalent to an acyclic constraint satisfaction problem of size $\|G\|^{O(i(G)\times hw(G))}$, where $i(\mathcal{G})$ is the intricacy of G and $hw(G)$ is the hypertree width of its strategic dependency hypergraph. Acyclic CSPs, in turn, are well-known to be solvable in polynomial time.

Moreover, [20] showed that in all above cases where a pure Nash Equilibrium can be computed in polynomial time, a Pareto Nash equilibrium can also be computed in polynomial time.

4.3 Congestion Games

[P6]. **INSTANCE:** A symmetric network congestion game.
OUTPUT: A pure Nash equilibrium.

COMMENT: [19] showed that there is a polynomial algorithm for finding a pure Nash equilibrium in symmetric network congestion games. This is found by computing the optimum of Rosenthal's potential function, through a reduction to min-cost flow.

Note that [19] only considers unit-demand users. Thus, unweighted single-commodity network congestion game are also a symmetric network congestion game. So, their result implies that *computing a pure Nash equilibrium for a unweighted single-commodity network congestion game is polynomially solvable*.

[P7]. **INSTANCE:** KP model (with varying user demands and uniformly related parallel machines).
OUTPUT: A pure Nash equilibrium.

COMMENT: [17] presented a polynomial-time algorithm that computes a a pure Nash equilibrium. Roughly speaking, the algorithm works in a greedy fashion; it considers each of the user traffics in non-increasing order and assigns it to the link that minimizes (among all links) the latency cost of the user had its traffic been assigned to that link. Clearly, the supports computed by the algorithm represent a pure strategy profile. They showed that this profile is a Nash equilibrium. This is showed by induction on the number of i iterations, $1 \leq i \leq n$, of the main loop of the algorithm. They proved that the system of users and links is in Nash equilibrium after each such iteration.

[P8]. INSTANCE: A (player-specific) matroid congestion game.
OUTPUT: A pure Nash equilibrium.

COMMENT: [3] showed, if the strategy space of each player in a congestion game consists of the bases of a matroid over the set of resources, then the lengths of all best response sequences are polynomially bounded in the number of players and resources. This yields a polynomial time algorithm for computing a pure Nash equilibrium of a matroid congestion game.

In addition, it is shown that the convergence result is tight, that is, the matroid property is a necessary and sufficient condition on the players' strategy spaces for guaranteeing polynomial time convergence to a Nash equilibrium.

The convergence result does not hold if players have player-specific latency functions. However, in [ARV06b] it is presented a polynomial time algorithm that computes a pure Nash equilibrium for any given player-specific matroid congestion game.

[P9]. INSTANCE: KP model with related links.
OUTPUT: A Nash equilibrium of minimum social cost.

COMMENT: [16] studied the problem of *Nashification*: Given any pure routing, the goal is to compute a Nash equilibrium with less or equal social cost. They present an $O(nm^2)$ time algorithm which nashifies any pure routing in the model of related link capacities. The routing problem considered here is equivalent to the scheduling problem for related machines. As an immediate consequence of our result, we get a PTAS for computing a Nash equilibrium with minimum social cost by applying the PTAS of Hochbaum and Shmoys [27] to the scheduling problem and nashifying the schedule.

One approach to nashify a routing would be to let the users, in some order, make greedy selfish steps until a Nash equilibrium is reached. [16] proved that for our routing problem there exists an instance of size polynomial in n such that the maximum length of a sequence of greedy selfish steps is at least $\Omega(2^{\sqrt{n}})$. This result is followed by an $O(2^n)$ upper bound for the length of any sequence of greedy selfish steps in the model of identical capacities. As a consequence they showed that nashifying a solution using the above mentioned approach may take time exponential in n.

This result immediately implies the existence of a PTAS for computing a pure Nash equilibrium of minimum social cost: first compute a configuration of social cost at most

$(1 + \epsilon)$ times the social optimum [27], and consequently transform it into a pure Nash equilibrium of at most the same social cost.

4.4 Games on Graphs

4.4.1 Virus Inoculation Game

[P10]. **Instance:** A virus inoculation game (G, C, L).
Output: A pure Nash equilibrium.

Comment: Aspnes et al. in [1] showed that Nash equilibria for the game are easily characterized in terms of the sizes of the components of the attack graph. The characterization states that there is always a component-size threshold $t = Cn/L$ such that each node will install the anti-virus software if it would otherwise end up in a component of vulnerable nodes with expected size greater than t, and will not install the software if it would otherwise end up in a component with expected size less than t. When the expected component size equals t, the node is indifferent between installing and not installing and may adopt a mixed strategy. The threshold arises in a natural way: it is the break-even point at which the cost C of installing the software equals the expected loss $L(t/n)$ of not installing.

The characterization implies that a pure Nash equilibrium always exists. Furthermore, using it they proved that a pure Nash equilibrium can be computed in $O(n^3)$ time and that any population of nodes will quickly converge to a Nash equilibrium by updating their strategies locally based on the other nodes' strategies.

4.4.2 Attackers-Defender Games

[P11]. **Instance:** An attackers-defender game $\langle \mathcal{N}, \{S_i\}_{i\in\mathcal{N}}, \{\mathsf{IP}\}_{i\in\mathcal{N}}\rangle$.
Output: A Nash equilibrium.

Comment: Mavronicolas et al. in [38, 39] showed first that the game posses no pure Nash equilibrium, unless the graph is trivial. Then they proceeded providing an elegant algebraic characterization of mixed Nash equilibria, which however, does not imply a polynomial time algorithm for their computation. However, in a subsequent work [37] they showed how to compute a (mixed) Nash equilibrium in polynomial time: by reducing it to a 2-players, zero sum-game. The latter is known to be solved in in polynomial time, using Linear programming [30].

Inspired by some graph-theoretic necessary conditions of Nash equilibrium discoved in [38, 39], the authors introduced a class of uniform Nash equilibrium, called *Matching Nash equilibria*. A matching profile enjoys some nice "covering" and "expanding" graph-theoretic properties. They showed:

[P12]. **INSTANCE:** An attackers-defender game $\langle \mathcal{N}, \{S_i\}_{i\in\mathcal{N}}, \{\mathsf{IP}\}_{i\in\mathcal{N}}\rangle$.
OUTPUT: A Matching Nash equilibrium.

COMMENT: Mavronicolas et al. in [37] first presented a characterization of graphs admitting Matching Nash equilibria. Then using it they developed a polynomial time algorithm which translates the characterization to (decide the existence of and) compute a Matching Nash equilibrium. This relies on obtaining a polynomial time algorithm for the (new) graph-theoretic problem of deciding, given a graph G, whether its *Independence Number* and *Edge Covering Number* $\beta'(G)$ are equal, and yielding, if so, a maximum independent set for the graph.

In an *Attacker Symmetric* and *Uniform Nash equilibrium* [37], all attackers have a common support on which each uses a uniform distribution.

[P13]. **INSTANCE:** An attackers-defender game $\langle \mathcal{N}, \{S_i\}_{i\in\mathcal{N}}, \{\mathsf{IP}\}_{i\in\mathcal{N}}\rangle$.
OUTPUT: An Attacker Symmetric and Uniform Nash equilibrium.

COMMENT: [37] provides a characterization of graphs admitting Attacker Symmetric and Uniform Nash equilibria. Roughly speaking, such graphs either allow the definition of a certain probability distribution on their edge set, or have their Independence Number equal to their Edge Covering Number, which implies that the graph admit Matching Nash equilibria. Thus, the characterization partitions the class of Attacker Symmetric Uniform Nash equilibria into two subclasses. Then they translate the characterization into a polynomial time algorithm to (decide the existence of and) compute an Attacker Symmetric Uniform Nash equilibrium.

---

5 Class $\mathcal{PLS}$

Here, we overview on problems for which their their computational complexity have been proven to be in the class $\mathcal{PLS}$.

5.1 Congestion Games

[P14]. **INSTANCE:** An unweighted congestion game.
OUTPUT: A pure Nash equilibrium.

COMMENT: [19] showed that it is $\mathcal{PLS}$-complete to find a pure Nash equilibrium in network congestion games of the following sorts:

- General congestion games.
- Symmetric congestion games.
- Asymmetric network congestion games.

This result means intuitively that the problem is "as hard to compute as any object whose existence is guaranteed by a potential function". Their proof implies the existence of examples with exponentially long shortest paths, as well as the $\mathcal{PSPACE}$-completeness of the problem of finding a Nash equilibrium reachable from a specified state. The completeness proof is complicated, as it requires the reworking of the reduction, to the problem of finding local optima of weighted MAX2SAT instances. When congestion games are posed in the abstract (in terms of sets of resources instead of paths in a network, this being the original formulation), Nash equilibria are $\mathcal{PLS}$-complete to find even in the symmetric case.

Their algorithm for pure Nash equilibria has an application to the non-atomic congestion games studied by Roughgarden and Tardos [52], in which delays are continuous functions. We show that, under the necessary smoothness assumptions, we can approximate the Nash equilibria of such games in strongly polynomial time.

[P15]. **INSTANCE:** A threshold congestion game.
OUTPUT: A Nash equilibrium.

COMMENT: [4] showed that the problem is $\mathcal{PLS}$-complete. Their proof is by a reduction from a local search version of MAXCUT with the *flip*-neighborhood. The local search version of MAXCUT can be described as a game, the so-called *party affiliation game* in which players correspond to nodes that can choose whether they belong to a set A or a set B. Edges reflect some symmetric kind of *anti-sympathy*, that is, a node seeks to choose the set A or B such that the weighted number of edges leading to the other set is maximized. Schäffer and Yannakakis [54] show that MAXCUT is $\mathcal{PLS}$-complete. Finally, they showed that MAXCUT can be reduced to threshold games.

The above result was used to show the following results:

[P16]. **INSTANCE:** A general (asymmetric) network congestion game with non-decreasing, linear delay functions.
OUTPUT: A Nash equilibrium.

COMMENT: [4] showed that the problem is $\mathcal{PLS}$-complete. Their proof uses quadratic threshold games (for which the problem is $\mathcal{PLS}$-complete) as the starting point for the $\mathcal{PLS}$-reduction.

5.2 Market Sharing Games

[P17]. **INSTANCE:** A market sharing game with polynomially bounded costs.
OUTPUT: A Nash equilibrium.

COMMENT: [4] showed that the problem is $\mathcal{PLS}$-complete. Their proof uses quadratic threshold games as the starting point for the $\mathcal{PLS}$-reduction.

5.3 Overlay Network Design Games

[P18]. **INSTANCE:** An overlay network design game.
OUTPUT: A Nash equilibrium.

COMMENT: [4] showed that the problem is $\mathcal{PLS}$-complete. Their proof uses quadratic threshold games as the starting point for the $\mathcal{PLS}$-reduction.

6 Class $\mathcal{PPAD}$

Here, we overview on problems for which their their computational complexity have been proven to be in the class $\mathcal{PPAD}$.

[P19]. **INSTANCE:** A game with r players.
OUTPUT: A Nash equilibrium.

COMMENT: [45] showed that the problem is $\mathcal{PPAD}$-complete. Their completeness proof uses a reduction to END OF THE LINE via Brouwer's Theorem and Sperner's Lemma.

[P20]. **INSTANCE:** A game with 4 players.
OUTPUT: A Nash equilibrium.

COMMENT: [13] showed that the problem is $\mathcal{PPAD}$-complete by reduction to the so called 3-DIMENSIONAL BROUWER problem, which is also proved to be $\mathcal{PPAD}$-complete in the same work. In particular, the proof in [13] is a reduction from a stylized version of Brouwer's problem, and utilizes certain "gadgets invented in a previous paper [24]: Games that behave like arithmetical and logical gates. The reduction then constructs a graphical game (a game whose players are nodes in a graph and who affect each other only if they are neighbors on the graph), combining the various gadgets, that ends up "computing" the value of a Brouwer function- in fact, averaged over a fine grid of points. Completeness of the 4-player case then follows from a result in [24] whereby any game can be simulated by a 4-player game. The number 4 comes up as the chromatic number of a particular undirected graph associated with a graphical game constructed in the proof of [13] (called here the moralized graph of the graphical game).

[P21]. **INSTANCE:** A game with r players, where r is a constant.
OUTPUT: A Nash equilibrium.

COMMENT: [24] first reduced the Nash problem for a d-graphical game, for fixed $d > 1$, to the same problem for a game with d^2 players in a normal form. The reduction is a polynomial one for graphical games with bounded degree (which is the intended use of graphical games).

They also give a reduction in the opposite direction: For any normal form game they construct a graphical game with all degrees bounded by three and with two strategies per player such that we can recover a Nash equilibrium of the original game from any Nash equilibrium of the graphical game. Notice that the degree 3 and two strategies restriction is significant here, since otherwise the reduction is immediate (via a graphical game played on an r-clique).

In fact, their reduction from r-player games to 4-player games is a composition of these two reductions: First they reduce the r-player game to a graphical game, then they reduce the graphical game to a normal form game, and obtain the 4-player result by specializing the latter reduction to the output of the former.

The reduction from graphical games to normal form games is based on a rather simple idea: color the graph, and simulate all vertices in a color class by a single player. This player represents the whole class by playing a mixed strategy that is the average of the mixed strategies played by the vertices in the class. In order to make sure that a color class player does not "neglect" any node by failing to include its strategies in its mixed strategy, they pair up the players, and have the pairs play against each other a generalization of Matching Pennies at very high-stakes: at any Nash equilibrium the color class player is now forced to assign the same probability mass to each vertex that it represents. The reduction from normal form games to graphical games is more sophisticated.

Finally, applying [13] result, i.e. that a 4-Nash problem is $\mathcal{PPAD}$-complete, the result [24] implies that r-Nash is $\mathcal{PPAD}$-complete.

[P22]. **INSTANCE:** A game with 3 players.
OUTPUT: A Nash equilibrium.

COMMENT: Daskalakis and Papadimitriou in [15] showed $\mathcal{PPAD}$completeness of the problem. Their proof is based on a variant of the arithmetical gadget of [24], containing few extra nodes. The new gadget, when used in the reduction from Brouwer's problem in [24], has the effect of creating a graphical game whose moralized graph is 3-colorable, thus reducing the number of players needed to simulate the graphical game to three.

Independently, Chen and Deng [6] have also come up with a proof of the same result. Their proof is quite different from ours: Without changing the gadgets, they come up with a more sophisticated version of the simulation in [24] in which just three players are needed to simulate a general game (despite the fact that 4 colors are required to color the moralized graph). They do this by exploiting a subtle form of disconnectedness in the graphical game constructed in [24].

[P23]. **INSTANCE:** A game with 2 players.
OUTPUT: A Nash equilibrium.

COMMENT: The $\mathcal{PPAD}$-completeness results for 3-Nash and 4-Nash [15, 13] results leave the two player Nash-equilibrium the last opening problem in the long sequel of search for an effcient solution. Chen and Deng in [7] settle the problem with a $\mathcal{PPAD}$-completeness proof for the 2-player Nash equilibrium problem. Their proof gets rid of the graphical game model and derived a direct reduction from 3-DIMENSIONAL BROUWER to 2-Nash. They designed new gadgets for various arithmetic and logic operations.

Daskalakis et al.in [12] defined a *bounded (division-free) straight-line program* to be an arithmetic binary circuit with nodes performing addition, subtraction, or multiplication on their inputs, or evaluating to pre-set constants, with the additional constraint that the values of all the nodes remain in $[0, 1]$. Given a succinct game, the following problem, called EXPECTED UTILITY, is of interest: Given a mixed strategy profile $x_1, \cdots, x_r$, compute the expected utility of player p. Using these notions [12] settled the $\mathcal{PPAD}$-completeness of succinct games:

[P24]. INSTANCE: A succinct game of polynomial type for which there is a bounded division free straight-line program of polynomial length for computing EXPECTED UTILITY.
OUTPUT: A Nash equilibrium.

COMMENT: Daskalakis et al. in [12] first showed that if for a succinct game $\mathcal{G}$ of polynomial type there is a bounded division free straight-line program of polynomial length for computing EXPECTED UTILITY, then $\mathcal{G}$ can be mapped in polynomial time to a graphical game $\mathcal{G}$ so that there is a polynomially computable surjective mapping from the set of Nash equilibria of $\mathcal{G}$ to the set of Nash equilibria of $\mathcal{G}$.

Then they used the mapping of the previous result (between succinct and graphical games) to deriving complexity results for the problem of computing a Nash equilibrium in succinct games. Actually, they shows that if for a succinct game $\mathcal{G}$ of polynomial type there is a bounded division free straight-line program of polynomial length for computing EXPECTED UTILITY, then the problem of computing a Nash equilibrium in the succinct game polynomially reduces to the problem of computing a Nash equilibrium of a 2-player game.

The latter result implies that the problem of computing a Nash equilibrium in the following families of succinct games can be polynomially reduced to the same problem for 2-player games: graphical games, congestion games, multimatrix games, semi-anonymous games, local effect games, scheduling games, hypergraphical games, network design games, and facility location games.

Finally, the recent proof of the $\mathcal{PPAD}$-completeness of 2-Nash [7] implies that computing a Nash equilibrium for in a succinct game of polynomial type for which there is a bounded division free straight-line program of polynomial length for computing EXPECTED UTILITY is $\mathcal{PPAD}$-complete. This implies also that computing a Nash equilibrium for the following families of succinct games is $\mathcal{PPAD}$-complete: graphical games, congestion games, multimatrix games, semi-anonymous games, local effect games, scheduling games, hypergraphical games, network design games, and facility location games.

---

7 Class $\mathcal{NP}$

Here, we overview on problems for which their their computational complexity have been proven to be in the class $\mathcal{NP}$.

[P25]. **Instance:** A strategic game in general form, where the number of actions of each a player is some constant k, for any $k \geq 2$.
Question: Does there exists a pure Nash equilibrium?

Comment: [2] showed that for strategic games in general (explicit) form, even for the case when the number of actions of each a player is some constant k, the problem of deciding whether the game admits a pure Nash equilibrium is $\mathcal{NP}$-complete. Their reduction utilizes the the Satisfiability of boolean formulae problem, which is known to be $\mathcal{NP}$-complete.

7.1 Symmetric Games

[P26]. **Instance:** A symmetric 2-player game.
Question: Does the game admit a Nash equilibrium with a certain property?

Comment: [10] showed that existence problems for various classes of Nash equilibria are $\mathcal{NP}$-hard: These are not the first results of this nature; most notably, Gilboa and Zemel provide some $\mathcal{NP}$-hardness results in the same spirit [25]. They provided a single reduction which in demonstrates (sometimes stronger versions of) most of their hardness results, and interesting new results. In particular they showed that even in symmetric 2-player games, it is $\mathcal{NP}$-hard to determine whether there exists a Nash equilibrium with any one of the following properties:

- all players have expected utility at least k, even when k is the largest number such that there exists a distribution over outcomes of the game such that all players have expected utility at least k.
- player 1 has expected utility at least k.
- player 1 sometimes plays a given $s_i \in \mathcal{S}_i$.
- player 1 sometimes never plays a given $s_i \in \mathcal{S}_i$.

Also, they showed that even in symmetric 2-player games, it is $\mathcal{NP}$-hard to determine whether there exists a Nash equilibrium there exists a Pareto-optimal Nash equilibrium.

[P27]. **Instance:** A restricted graphical $(\mathcal{G}, \mathsf{M})$ game.
Question: Does the game admit a pure Nash equilibrium?

Comment: [20] showed that for the following restricted cases of graphical games the problem is $\mathcal{NP}$-complete:

- Games with a bounded neighborhood
- Acyclic-graph games, and acyclic-hypergraph games.

Hardness holds even if the game d has 3-bounded neighborhood, and the number of actions is fixed. Moreover, the same results hold for Pareto pure Nash equilibria.

[P28]. **Instance:** A acyclic-hypergraph game that has small neighborhood or a graphical game.
Question: Does the game admit a strong Nash equilibrium?

Comment: [20] showed that for the problem is $\mathcal{NP}$-complete for such games. Hardness holds even if $\mathcal{G}$ is a graphical games and has 3-bounded neighborhood.

7.2 Congestion Games

[P29]. **Instance:** KP model.
Output: A Nash equilibrium with the *minimum social cost* for the given system of users and links.

Comment: The problem was shown to be $\mathcal{NP}$-hard in [17]. Their reduction reduces from BIN PACKING. Using similar methodology they also prove for the same model, computing a pure Nash equilibrium with the *maximum social cost* for the given system of users and links is $\mathcal{NP}$-hard.

[P30]. **Instance:** An arbitrary static coalitional congestion game over identical parallel links among players with integer weights.
Output: A pure Nash equilibrium.

Comment: Fotakis et al. in [18] showed that the problem is $\mathcal{NP}$-complete, even if we enforce a number of coalitions $k = 1 + n(1 - \delta)$, for any constant $\delta \in (0, 1]$. Intuitively, this means that the problem remains $\mathcal{NP}$-hard, even if we demand a very large number of coalitions of players, so long as we allow one coalition have large cardinality.

Note that the authors showed existence of pure Nash equilibria in coalitional congestion games, even if we assume unrelated parallel links and we allow the players to form *dynamic coalitions* (i.e., consider the case of arbitrary combinations of players that attempt a joint selfish move). This is done by showing the existence of a generalized ordinal potential function which assures the convergence to a pure Nash equilibrium in a finite number of steps.

For the case of unrelated parallel links, they proved that any improvement path that combines selfish movements of (even dynamically forming) coalitions, has length at most $2^{W_{tot}}$ where here (for the more general case of unrelated parallel links) $W_{tot} = \sum_{j\in[n]} max_{l\in[m]}\{w_j(l)\}$.

7.3 Games on Graphs

7.3.1 Virus Inoculation Game

[P31]. **INSTANCE:** A virus inoculation game (G, C, L).
OUTPUT: A pure Nash equilibrium.

COMMENT: Aspnes et al. in [1] showed computing the pure Nash equilibrium with lowest cost and computing the pure Nash equilibrium with highest cost are $\mathcal{NP}$-hard problems. Their completeness proof reduces from the VERTEX COVER.

Moreover, they have shown how a near-optimal deployment of anti-virus software can be computed by reduction to the SUM-OF-SQUARES PARTITION problem, a new variant of classical graph partitioning problems where the goal is to remove m vertices so as to minimize the sum of the squares of the sizes of surviving components. Though it is $\mathcal{NP}$-hard to solve this problem exactly, they gave a polynomial-time $O(log^2 n)$-approximation algorithm for sum-of-squares partition, which yields a corresponding approximation algorithm for anti-virus software deployment. This algorithm may be of use as a network administration tool for choosing how to deploy anti-virus software to minimize the combined costs of deployment and infection and as a public-health tool for designing inoculation strategies for containing outbreaks of highly-infectious diseases when a good approximation to the interaction graph can be computed but the initial source of contagion is unknown.

7.3.2 Attackers-Defender Games

In a *Defender Uniform Nash equilibrium* [37], the defender chooses each edge in its support with uniform probability.

[P32]. **INSTANCE:** An attackers-defender game $\langle \mathcal{N}, \{S_i\}_{i\in\mathcal{N}}, \{\mathsf{IP}\}_{i\in\mathcal{N}}\rangle$.
OUTPUT: A Defender uniform Nash equilibrium.

COMMENT: [37] provides a characterization of graphs admitting Defender uniform Nash equilibria. The characterization involves *Regular Subgraphs* and Independent Sets. Then, they prove that deciding the existence of a Defender Uniform Nash equilibrium is $\mathcal{NP}$-complete. This employs a reduction from the UNDIRECTED PARTITION INTO HAMILTONIAN SUBGRAPHS OF SIZE AT LEAST SIX problem, which is proved $\mathcal{NP}$-complete in the same work. The latter completeness proof reduces from the directed version of the problem for subgraphs of size at least 3.

---

8 Class $\mathcal{NEXP}$

Here, we overview on problems for which their their computational complexity have been proven to be in the class $\mathcal{NEXP}$.

[P33]. **INSTANCE:** A d-dimensional torus game.
QUESTION: Does the game have a pure Nash equilibrium?

COMMENT: [14] showed a dichotomy regarding pure Nash equilibria for d-dimensional games: For 1-dimensional torus (i.e. a cycle) game [14] showed that we can check whether the game has a pure Nash equilibrium in polynomial time. The algorithm for the cycle is based on a rather sophisticated analysis of the cycle structure of the Nash dynamics of the basic game. However, the problem for $d \geq 2$ the problem of deciding whether there exists a pure Nash equilibrium in a fully symmetric d-dimensional torus game becomes $\mathcal{NEXP}$-complete. The $\mathcal{NEXP}$-completeness is established by a generic reduction which, while superficially quite reminiscent of the tiling problem [35], relies on several novel tricks for ensuring faithfulness of the simulation.

[P34]. **INSTANCE:** A d-dimensional torus game.
OUTPUT: A succinct description of a mixed Nash equilibrium.

COMMENT: [14] For any $d \geq 2$, we can compute a succinct description of a mixed Nash equilibrium of a d-dimensional torus game in deterministic exponential time, in particular polynomial in 2^s, the size of the game description and the number of bits of precision required, but independent of the number of players.

---

9 Class #P

Here, we overview on problems for which their their computational complexity have been proven to be in the class #P.

[P35]. **INSTANCE:** A symmetric 2-player games.
OUTPUT: The number of Nash equilibria.

COMMENT: [10] showed that counting the number of Nash equilibria is #P-hard.

10 Polynomial Hierarchy

Here, we overview on problems for which their their computational complexity have been proven to be in the Polynomial Hierarchy class.

[P36]. **INSTANCE:** A strategic game.
QUESTION: Does there exists a strong Nash equilibrium?

COMMENT: [20] showed that the problem is Σ_2^P-complete and thus at the second level of the Polynomial Hierarchy. Hardness holds even if the game is a graphical one and has 3-bounded neighborhood, and the number of actions is fixed. The proof of this theorem

provides a fresh game-theoretic view of the class Σ_2^P as the class of problems whose positive instances are characterized by a coalition of players who cooperate to provide an equilibrium, and win against any other disjoint coalition, which fails in trying to improve the utility for all of its players. E.g., in the case of Σ_2 quantified Boolean formulas, the former coalition consists of the existentially quantified variables, and the latter of the universally quantified ones.

[P37]. **INSTANCE:** A strategic game in implicit form.
QUESTION: Does there exists a pure Nash equilibrium?

COMMENT: [2] showed that given a strategic game in implicit form, the problem of deciding whether the game admits a pure Nash equilibrium is Σ_2^P-complete. Their reduction utilizes the Q2SAT, which is known to be Σ_2^P-complete.

The proof of this result uses a game with four players, for the sake of clarity, but a similar game with only two players can also be used. This implies that, the problem of deciding whether a game in implicit form with k players, for any $k \geq 2$, admits a pure Nash equilibrium is Σ_2^P-complete.

11 Polynomial Space

Here, we overview on problems for which their their computational complexity have been proven to be in the Polynomial Space class.

[P38]. **INSTANCE:** A strategic game.
QUESTION: Does there exists a pure Nash equilibrium?

COMMENT: [20] showed that the following existence problems are all feasible in logarithmic space:

- for a pure Nash equilibrium
- for a pure Pareto equilibrium
- for a strong Nash equilibrium (and computing all such equilibria).

Acknowledgments. We would like to thank Heiner Ackermann, Spyros Kontogiannis and Berthold Vöcking for their valauble contributions to the DELIS deliverable 4.1.3, which formed the backbone for this survey.

References

[1] James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation Strategies for Victims of Viruses and the Sum-of-squares Partition Problem. *Journal of Computer and System Sciences*, 72(6):1077–1093, September 2006.

[2] C. Àlvarez, J. Gabarró, and M. J. Serna. Pure Nash Equilibria in Games with a Large Number of Actions. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 95–106. Springer, September 2005.

[3] H. Ackermann, H. Röglin, and B. Vöcking. Pure Nash Equilibria in Player-Specific and Weighted Congestion Games. In *Proceedings of the 2nd International Workshop on Internet and Network Economics*, volume 4286 of *Lecture Notes in Computer Science*. Springer, 2006.

[4] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the Impact of Combinatorial Structure on Congestion Games. *Journal of the ACM*, 55(6):1–22, 2008.

[5] R. Aumann. Acceptable Points in General Cooperative n-Person Games. *Contributions to the Theory of Games IV*, 40:287–324, 1959.

[6] X. Chen and X. Deng. 3-Nash is $\mathcal{PPAD}$-Complete. Technical Report TR05-134, *Electronic Colloquium in Computational Complexity*, 2005.

[7] X. Chen and X. Deng. Settling the Complexity of 2-Player Nash-Equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 261–272. IEEE Press, 2006.

[8] V. Conitzer, J. Lang Lang, and T. Sandholm. How Many Candidates are Needed to Make Elections Hard to Manipulate? In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 201–214, June 2003.

[9] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM Press, May 1971.

[10] V. Conitzer and T. Sandholm. New Complexity Results About Nash Equilibria. *Games and Economic Behavior*, 63(2):621–641, July 2008.

[11] G.B. Danzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[12] C. Daskalakis, A. Fabrikant, and C. H. Papadimitriou. The Game World Is Flat: The Complexity of Nash Equilibria in Succinct Games. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, volume 4051 of *Lecture Notes in Computer Science*, pages 513–524. Springer, July 2006.

[13] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 71–78, May 2006.

[14] C. Daskalakis and C. H. Papadimitriou. The Complexity of Equilibria in Highly Regular Graph Games. In *Proceedings of the 13th Annual European Symposium on Algorithms*, volume 3669 of *Lecture Notes in Computer Science*, pages 71–82. Springer, October 2005.

[15] C. Daskalakis and C. H. Papadimitriou. Three-Player Games are Hard. Technical Report TR05-139, *Electronic Colloquium in Computational Complexity*, 2005.

[16] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the Coordination Ratio for a Selfish Routing Game. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming,*, volume 2719 of *Lecture Notes in Computer Science*, pages 514–526. Springer, June 2003.

[17] D. Fotakis, S. C. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. *Theoretical Computer Science*, 343(1-2):123–134, October 2005.

[18] D. Fotakis, S. C. Kontogiannis, and P. G. Spirakis. Atomic Congestion Games Among Coalitions. *ACM Trans. Algorithms*, 4(4):1–27, May 2008.

[19] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The Complexity of Pure Nash Equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 604–612. ACM Press, June 2004.

[20] G. Gottlob, G. Greco, and F. Scarcello. Pure Nash Equilibria: Hard and Easy Games. *Journal of Artificial Intelligence Research*, 24:357–406, 2005.

[21] I. Gilboa. The Complexity of Computing Best Response Automaton in Repeated Games. *Journal of Economic Theory*, 45(2):342–352, 1988.

[22] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[23] M. Goemans, E. Li, V. Mirrokni, and M. Thottan. Market Sharing Games Applied to Content Distribution in Ad-Hoc Networks. In *Proceedings of the 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*, pages 55–66. ACM, May 2004.

[24] P. W. Goldberg and C. H. Papadimitriou. Reducibility Among Equilibrium Problems. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 61–70. ACM Press, May 2006.

[25] I. Gilboa and E. Zemel. Nash and Correlated Equilibria: Some Complexity Considerations. *Games and Economic Behavior*, 1:80–93, July 1989.

[26] J. T. Howson. Equilibria of Polymatrix Games. *Management Science*, 18(5):312–318, January 1972.

[27] D. S. Hochbaum and D. B. Shmoys. A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach. *SIAM Journal of Computing*, 17(3):539–551, June 1988.

[28] III J. J. Bartholdi and J. B. Orlin. Single Transferable Vote Resists Strategic Voting. *Social Choice and Welfare*, 8:341–354, October 1991.

[29] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How Easy is Local Search? *Journal of Computer and System Sciences*, 37(1):79–100, August 1988.

[30] L.G. Khachian. A Polynomial Algorithm in Linear Programming. *Doklady Akademii Nauk, English translation in Soviet Doklady Mathematics*, 20:191–194, 1979.

[31] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated Equilibria in Graphical Games. In *Proceedings of the 4th ACM Annual Conference on Electronic Commerce*, pages 42–47, 2003.

[32] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell System Technical Journal*, 29:309–322, January 1970.

[33] M. Kearns, M. Littman, and S. Singh. Graphical Models for Game Theory. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 253Ű–260, Aug 2001.

[34] M. Kearns and L. Ortiz. Algorithms for Interdependent Security Games. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, pages 288–297. MIT Press, December 2004.

[35] H. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 2nd edition, 1997.

[36] D. Lichtenstein and M Sipser. GO is Polynomial-space Hard. *Journal of the ACM*, 27(2):393–Ű401, April 1980.

[37] M. Mavronicolas, L. Michael, V. G. Papadopoulou, A. Philippou, and P. G. Spirakis. The Price of Defense. In *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science)*, volume 4162 of *Lecture Notes in Computer Science*, pages 717–728. Springer, 2006.

[38] M. Mavronicolas, V. G. Papadopoulou, A. Philippou, and P. G. Spirakis. A Network Game with Attacker and Protector Entities. In *Proceedings of the 16th Annual International Symposium on Algorithms and Computation*, volume 3827 of *Lecture Notes in Computer Science*, pages 288–297. Springer, 2005.

[39] M. Mavronicolas, V. G. Papadopoulou, A. Philippou, and P. G. Spirakis. A Network Game with Attackers and a Defender. *Algorithmica*, 51(3):315–341, May 2008. Special Issue with selected papers from the 16th Annual International Symposium on Algorithms and Computation.

[40] O. Morgenstern and J. von Neumann. *The Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[41] J. F. Nash. Equilibrium Points in N-Person Games. In *Proceedings of National Acanemy of Sciences of the United States of America*, volume 36, pages 48–49, 1950.

[42] J. F. Nash. Non-cooperative Games. *Annals of Mathematics*, 54(2):286–295, 1951.

[43] G. Owen. *Game Theory*. Academic Press, 1995. Third Edition.

[44] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[45] C. H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, June 1994.

[46] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing*, pages 749–753. ACM Press, 2001.

[47] C. H. Papadimitriou. Computing Correlated Equilibria in Multi-Player Games. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 49–56. ACM Press, 2005.

[48] C. H. Papadimitriou and T. Roughgarden. Computing Equilibria in Multi-Player Games. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–91, 2005.

[49] C. H. Papadimitriou, A. A. Schaffer, and M. Yannakakis. On the Complexity of Local Search. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 438–445. ACM Press, May 1990.

[50] R. W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *International Journal of Game Theory*, 2(65–67), December 1973.

[51] N. Robertson and P. Seymour. Graph Minors -II: Algorithmic Aspects of Treewidth. *Journal of Algorithms*, 7(3):309–322, September 1986.

[52] T. Roughgarden and E. Tardos. Bounding the Inefficiency of Equilibria in Nonatomic Congestion Games. *Games and Economic Behavior*, 47(2):389–403, 2004.

[53] L. Stockmeyer and A. Meyer. Word Problems Requiring Exponential Time. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, pages 1–9, May 1973.

[54] A. A. Schäffer and M. Yannakakis. Simple Local Search Problems That are Hard to Solve. *SIAM Journal of Computing*, 20(1):56–87, February 1991.