# Compact game representations in graph for computing Nash equilibria

Lin Xu
xulin730@cs.ubc.ca

Shuang Hao
haos05@cs.ubc.ca

Department of Computer Science, University of British Columbia,
Vancouver, B.C. V6T 1Z4

December 10, 2005

**Abstract**

Nash equilibrium is one of the most important concepts in game theory. A lot of research efforts are invested into finding Nash equilibria of games. Unfortunately, all existing methods for finding Nash equilibria of arbitrary games require exponential runtime. This makes the application of Nash equilibria impractical. Real world games often have topological independencies. An agent's payoff does not always rely on all other agents' actions; most likely it only depends on a small group of agents' actions. In this paper, we introduce two well-known types of compact game representations: agent graph game and action graph game. The recently proposed state of the art method, the continuation method, is applied to these two types of compact games. Computing the Jacobian of the pay-off function is the bottleneck step of finding Nash equilibria. Theoretical analysis shows exponential savings can be achieved by exploiting the structure of compact game representation. Furthermore, we show that for an important sub-class of action graph game, symmetric game, the jacobian can be computed in polynomial time.

**Keywords:** Nash equilibria, Continuation method, Graphical games, Action-graph games

## 1 Introduction

Nash equilibrium is one of the central conceptions in Game Theory. In an $n$-player game, given a strategy profile $s = (s_1, \dots, s_n)$, we define $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$, then $(s_i, s_{-i}) = s$ [1]. Player $i$'s best response to the strategy profile $s_{-i}$ is defined as the strategy $s_i^* \in S_i$ such that its utility $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all strategies $s_i \in S_i$. A Nash equilibrium is a strategy profile $s = (s_1, \dots, s_n)$, if for all agents $i$, $s_i$ is its best response to $s_{-i}$. For approximate Nash equilibria, we relax the restriction of the best response to $u_i(s'_i, s_{-i}) + \epsilon \geq u_i(s_i, s_{-i})$, where $s_{i'}$ is the $\epsilon$-best response to $s_{-i}$. An $\epsilon$-Nash equilibrium is a strategy profile such that any player's strategy is its

$\epsilon$-best response to other players' strategies. Thus, no player can improve their expected payoff by more than $\epsilon$ by deviating from an approximate Nash equilibrium [2].

Nash's theorem reveals that *every finite normal form game has a Nash equilibrium*. A natural problem then is how to find out all(or some) Nash equilibria for a certain game. The two-player, two-action game is easy, as we can represent the utilities of agents in a tabular form and explore the strategy space by enumeration. But for the general case, the strategy space is exponential in the number of agents in the game, making the problem intractable. It is still not clear whether finding a Nash equilibrium belongs to $\mathcal{P}$. The best known algorithm so far to compute Nash equilibria is the continuation method due to Govindan and Wilson [6]. Their method guarantees to find at least one equilibrium of a game. However, in practice this algorithm's runtime is dominated by the computation of the Jacobian of the payoff function, required for computing the gradient, which is both best- and worst-case exponential in the number of agents.

In standard game representations, both the normal (matrix) form and the extensive (game tree) form obscure certain important structure that is often present in real-world scenarios. Compact models are required to make such structures explicit. Hence, we can utilize the utility independencies to reduce the strategy space, and alleviate the computation cost for finding Nash equilibria. Graphical models are widely used to compactly represent a game because they efficiently encode context-specific and strict utility independencies. Such typical models include graphical games, multi-agent influence diagrams, action-graph games etc.

The rest of the paper is organized as follows. We first introduce two kinds of graph representations for games in section 2. In section 3, we mainly illustrate the mechanism of the continuation method. Section 4 demonstrates how to apply the continuation method to compact games. Finally we draw conclusions in section 5 with some discussions.

## 2 Graphical models to represent a game

We explore two classes of graphical models for compactly representing games. The first one, agent-graph game, exploits strict independencies between players' utility functions. The second one, action-graph game, not only depicts strict independencies but also expresses context-specific independencies between players' utility functions.

### 2.1 Agent-graph games

An $n$-player graphical game is a pair $(G, u)$ [2], where $G$ is an undirected graph on $n$ vertices and $u$ is a set of $n$ matrices $u_i$. $u_i(\mathbf{a})$ specifies the utility to player $i$ when the joint action of the $n$ players is **a**. An edge $(i, j)$ in $G$ represents the fact that $i$'s utility is dependent on $j$'s action/decision, or vice versa. A mixed strategy for player $i$ is $s_i \in \pi(\mathbf{a}_i)$, a distribution over $i$'s possible actions. We use $N_G(i) \subseteq \{1, \ldots, n\}$ to denote the set of neighbors of player $i$ in $G$. Then $u_i$ is a $|N_G(i)|$-dimention matrix, where each index indicates one of its neighbors, including itself. The expected utility for player $i$ given a strategy profile $s$ is shown below:

$$V_i(s) = \sum_{a' \in \mathbf{a}_{N_G(i)}} u_i(a') \prod_{j \in N_G(i)} s_j(a'_j) \tag{1}$$

where $\mathbf{a}_{N_G(i)}$ is the action set for $i$'s neighbors. That reflects the strict utility independence held between pairs of players. The best response for player $i$ to the strategy profile $s_{-i}$ is a mixed strategy $s_i^*$, such that $\forall s_i \in \mathbf{s}_i$, $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$. In a Nash equilibrium, the strategy for every player $i$ is its best response to other players' strategy profile. We give a simple example in Figure 1. Player $a$'s utility only depends on the chosen actions of $a, b, c, g, f$, which are in the neighborhood of $a$.

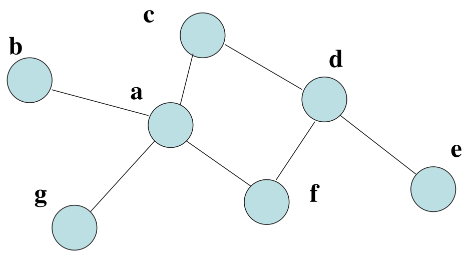


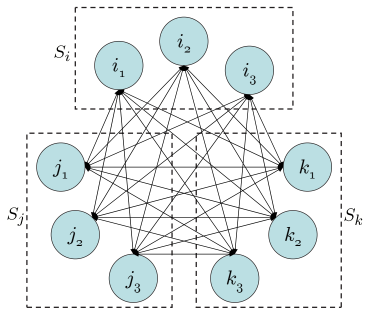
Figure 1: A simple example of graphical game

This definition alters nothing about the underlying game theory. Thus, every graphical game has a Nash equilibrium. Note that $N_G(i) << n$ makes the description of the game exponential in the degree of the graph but not in the total number of agents , which will significantly reduce the computation cost of finding Nash equilibria.

Such a graphical model may naturally and succinctly capture the underlying game structure in many common settings. A 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 the game involving their direct reports and their own managers, and so on up to the CEO. A graph may coincide with the topology of a computer network, in which each machine negotiates with its neighbors (to balance load, for instance).

## 2.2 Action-graph games

A novel graphical model, action-graph game [3], is proposed to depict detailed compact information, where the vertices represent actions rather than agents. An action-graph game (AGG) is a tuple $(N, \mathbf{S}, S, v, u)$, where $N$ is the set of $n$ agents; $\mathbf{S} = \prod_{i \in N} S_i$ is the set of pure action profiles; $S = \bigcup_{i \in N} S_i$ is the set of distinct actions. The neighbor relation $v$ is a function $v : S \to 2^S$; Utility function $u : S \times \Delta \to \mathbb{R}$, where $\Delta$ denotes the set of possible distributions of agents' numbers over distinct actions. Note that all agents have the same utility function.



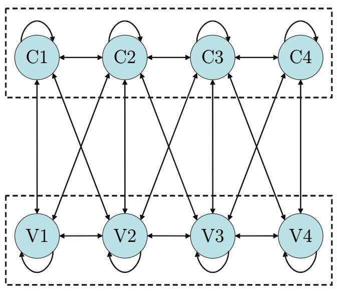


Figure 2: 3-player, 3-action game in AGG  Figure 3: ice cream vendor game in AGG

Let $G$ be the action graph: a directed graph having one node for every action $s \in S$. There is an edge from $s'$ to $s$ in $G$ if and only if $s' \in v(s)$. The utility function has the property that given any action $s$ and any pair of distribution $D$ and $D' \in \Delta$,

$$[\forall s' \in v(s), D(s') = D'(s')] \Longrightarrow u(s, D) = u(s, D') \tag{2}$$

That means player $i$'s utility depends only on the numbers of agents who take neighboring actions. So AGG is able to express context-specific independencies in utility function and strict utility independence.

It is illustrated in [3] that graphical games can be converted to AGGs. Figure 2 demonstrates that any game can be expressed as an AGG, where nodes represent actions; directed edges represent membership in a node's neighborhood; the dotted boxes represent the players' action sets: player 1 has actions 1, 2 and 3.

In Figure 3, AGGs are able to represent context-specific independence. Consider the setting in which $n$ ice cream vendors must choose one of four locations along a beach. Vendors sell either chocolate or vanilla ice cream, but not both. Chocolate (vanilla) vendors are negatively affected by the presence of other chocolate (vanilla) vendors in the same or neighboring locations, and are simultaneously positively affected by the presence of nearby vanilla (chocolate) vendors. Note that this game exhibits context-specific independence without any strict independence, and that the graph structure is independent of $n$.

$\Sigma_i = \pi(S_i)$ is the set of mixed strategies for $i$. $\Sigma = \prod_{i \in N} \Sigma_i$ is the strategy space for the game. We denote an element of $\Sigma_i$ by $\sigma_i$ and an element of $\Sigma$ by $\sigma$. Define the expected utility to agent $i$ for playing pure strategy $s$, given that all other agents play the mixed strategy profile $\sigma_{-i}$,

$$V_s^i(\sigma_{-i}) = \sum_{s-i \in S_{-i}} u(s, s_{-i}) \mathrm{Pr(s_{-i} \mid \sigma_{-i})} \tag{3}$$

The set of $i$'s pure strategy best responses to a mixed strategy profile $\sigma_{-i}$ is $\arg\max_s V_s^i(\sigma_{-i})$, so the full set of $i$'s pure and mixed strategy best responses to $\sigma_{-i}$ is

$$BR_i(\sigma_{-i}) \equiv \pi(\arg\max_s V_s^i(\sigma_{-i})) \tag{4}$$

A Nash equilibrium is the strategy profile $\sigma$, where $\forall\, i \in N, \sigma_i \in BR_i(\sigma_{-i})$. Note that the definition for best responses is slightly different with the formal definition [1], but such adaption will not affect the Nash equilibrium.

Here, we introduce the process *projection*, which will be used later. For every action $s \in S$ define a reduced graph $G^{(s)}$ by including only the nodes $v(s)$ and a virtual node $\phi$. The only edges included in $G^{(s)}$ are the directed edges related to $s$ and $s' \in v(s)$. The distribution $D^{(s)}$ is defined below.

$$D^{(s)}(s') = \begin{cases} D(s') & s' \in v(s) \\ \sum_{s'' \notin v(s)} D(s'') & s' = \phi \end{cases} \tag{5}$$

By the projection, we map the action nodes that are not in $i$'s neighborhood into a single action $\phi$.

## 3 Approaches to compute Nash equilibria

### 3.1 Related works

Early algorithms for computing Nash equilibria in graphical games have constraints on the graph's topology. For example, the abstract tree algorithm [2] requires the underlying graph to be a tree. Several other algorithms have been proposed for only computing the $\epsilon$-equilibria in graphical games, such as variable elimination [4]. Unfortunately, finding an $\epsilon$-equilibrium is not necessarily a step toward finding an exact equilibrium: the fact that $\sigma$ is an $\epsilon$-equilibrium does not guarantee the existence of an exact equilibrium in the neighborhood of $\sigma$ [5]. As stated above, it is not clear whether computing a Nash equilibrium belongs to $\mathcal{P}$, so many researchers devote their efforts to finding more efficient algorithms to solve this problem.

One state of the art general-purpose algorithm is the continuation method by Govindan and Wilson [6], a gradient-following algorithm which is based on topological insight into the graph of the Nash equilibrium. Although this algorithm impressively outperforms other algorithms, it is only practical when there are only small number of players and possible actions.

### 3.2 The continuation method

The continuation method works by solving a simpler perturbed problem and then tracing the solution as the magnitude of the perturbation decreases, converging to a solution to the original problem. When $\lambda = 1$, the perturbed problem's solution is known or easy to compute. When $\lambda = 0$, the perturbed problem is the original problem. The easily solvable problem can be "continuously deformed" into the target problem. Let vector $\omega$ represents the real values of the solution. With different values of $\lambda$, we obtain different perturbed problems. For every perturbed problem, we characterize solutions by the equation $F(\omega, \lambda) = 0$, where $F$ is a real value vector function. We define this function as follows: If $F(\omega, \lambda) = 0$ holds, $\omega$ is a solution to the problem perturbed by $\lambda$. The continuation method traces solutions along the manifold of solution pair $(\omega, \lambda)$ of that function $F(\omega, \lambda) = 0$. If we have a solution pair $(\omega, \lambda)$, our goal is to trace that solution to adjacent solutions. This requires the effects caused by $\lambda$ and $\sigma$ must cancel out so that $F$ remains equal to $0$. As we decrease $\lambda$ by $d\lambda$, the effect to the function $F(\omega, \lambda)$ must be cancel out by changing $\omega$ by $d\omega$, which is equivalent to

$$[\nabla_\omega F \quad \nabla_\lambda F] \begin{bmatrix} d\omega \\ d\lambda \end{bmatrix} = 0 \tag{6}$$

If the matrix $[\nabla_\omega F \quad \nabla_\lambda F]$ has null-space of rank $1$ everywhere, the curve is uniquely defined. If we construct function $F$ carefully, we can guarantee the curve starting with $\lambda = 1$ will across to $\lambda = 0$, thus finding the solution of the original problem. The null-space of the Jacobian $\nabla F$ at current solution $(\omega, \lambda)$ indicates a direction, along which the solution is moved step by step. Beginning with $\lambda = 1$, the continuation method can follow this path until $\lambda$ reaches $0$. The cost of each step is cubic in the size of the Jacobian.

The continuation method can be applied to the task of finding Nash equilibria in games. We perturb the game by adding $\lambda$ times a fixed bonus **b** to each agent's payoff, such that the bonus for an agent only depends on its own action. If the bonus given to agent $i$ for playing action $s_i$ is sufficiently large, while the bonus for agent $i$'s other actions are zero. Then the dominant strategy for each agent will be to play the action with the largest bonus. The corresponding strategy profile will form a Nash equilibrium, which is the only pure strategy Nash equilibrium for the perturbed game with $\lambda = 1$. By the continuation method we can follow a path in the space of $\lambda$ and equilibrium profiles for the resulting perturbed game. When $\lambda$ decreases to $0$, the corresponding strategy profile is a Nash equilibrium of the original game.

To apply the continuation method to find Nash equilibra of a game, we need to construct the function $F$, which has solution pair $(\omega, \lambda)$, where $\omega$ is the equilibrium of the perturbed game controlled by parameter $\lambda$. We define a retraction function $R : \mathbb{R}^m \to \Sigma$ to be an operator taking a vector of dimension $m$ and mapping it to the nearest point in the space of $\Sigma$ of mixed strategies in Euclidean distance. This mapping corresponds to increasing the probabilities of playing strategies that have better payoffs and decreasing the probabilities of playing strategies that have worse payoffs. A Nash equilibrium is recovered from $\sigma + V(\sigma)$ by the retraction operator $R : \ R(\sigma + V(\sigma)) = \sigma$, where no further (local) improvement can be achieved for any agent. Recall that $V$ is the vector derived form Equation (3), which is the expected utility to agent $i$ for playing pure strategy $s$, given that all other agents play the mixed strategy profile $\sigma_{-i}$. If $\sigma = R(\omega)$ and $\omega = \sigma + V(\sigma)$ we have the equivalent condition that $\omega = R(\omega) + V(R(\omega))$. This allows us to search for a point $\omega \in \mathbb{R}^m$ which satisfies this equality, in which case $R(\omega)$ is guaranteed to be an equilibrium.

In the perturbed game, we replace $V$ with $V + \lambda\mathbf{b}$, where **b** is a unique bonus that agents receive for playing certain actions no matter what all other agents do. If **b** is large enough, the resulting perturbed game has a unique pure strategy Nash equilibrium, in which each agent plays the pure strategy s for which $b_s$ is maximal. Our continuation equation has the form:

$$F(\omega, \lambda) = \omega - R(\omega) - (V(R(\omega)) + \lambda\mathbf{b}) \tag{7}$$

$F(\omega, \lambda) = 0$ if and only if $R(\omega)$ is an equilibrium of the induced game with bonus

$\lambda$**b**. When $\lambda = 1$, the solution of the perturbed game is the Nash equilibrium described above. When we decrease $\lambda$ to $0$, the perturbed game will be the same as the original game. Hence $F(\omega, 0) = 0$, if and only if $R(\omega)$ is a Nash equilibrium of the original game.

In order to solve Equation (6), we need to compute $\nabla_\omega F$. From Equation (7), we obtain

$$\nabla_\omega F = I - (I + \nabla V)\nabla R \tag{8}$$

The computation cost of computing the Jacobian of $F$ is dominated by the Jacobian of $V$. For pure strategy $s_i$ of agent $i$ and $s_{i'}$ of agent $i'$, the value of Jacobian at location $(s_i, s_{i'})$ equals to the expected payoff of agent $i$ when it plays the pure strategy $s_i$, and agent $i'$ plays the pure strategy $s_{i'}$ and all other agents play according to the strategy profile $\sigma$.

$$\nabla V^{i,i'}_{s_i,s_{i'}} = \sum_{\bar{s}\in\bar{S}} u(s_i, s_{i'}, \bar{s}) Pr(\bar{s} \mid \bar{\sigma}) \qquad \text{where } Pr(\bar{s} \mid \bar{\sigma}) = \prod_{j\in\bar{N}} \bar{\sigma}_j(\bar{s}_j) \tag{9}$$

We denote $\bar{\sigma} \equiv \sigma_{-\{i,i'\}}$ as the strategy profile for agents other than $i$ and $i'$. In the rest part of the paper, we will use this kind of notation in similar cases.

The computing cost of this Jacobian (multiplications and summations in Equation (9)) is exponential in the number of agents.

## 4 Applying continuation method to structured games

In order to minimize the computation cost of finding the Nash equilibria of a game using the continuation method, we have to reduce the cost of computing the Jacobian of $\nabla V$, which requires exponential time in the number of agents. In this section, we show that using compact representation of games can significantly reduce the computation cost of computing the Jacobian by exploiting topological structure.

### 4.1 Computing the Jacobian in agent-graph games

In graphical games, agent $i$'s payoff only depends on agent $i$'s action and its neighbors' actions. This observation allows us to achieve an exponential saving on the computation cost of the Jacobian.

When agent $i'$ is not in the neighborhood of agent $i$, the equation ignores the action of $i'$. The computation is exponential only in the neighborhood size of agent $i$. We also observe the result of the equation has nothing to do with agent $i'$ and its actions. Hence, we can use this value for all agents not in the neighborhood of $i$.

$$\begin{aligned} \nabla V^{i,i'}_{s_i,s_{i'}} \quad &= \quad \sum_{s_{-i}\in S_{-i}} u(s_i, s_{-i}) Pr(s_{-i} \mid \sigma_{-i}) \qquad (10) \\ &\text{where } Pr(s_{-i} \mid \sigma_{-i}) = \prod_{j\in N_G(i)-\{i\}} \sigma^j_{-i}(s^j_{-i}) \end{aligned}$$

Recall that $S_{-i}$ is a vector of actions only of $N_G(i) - \{i\}$, we see that this computation is only in the neighborhood size of $i$. Since $i'$ never explicitly appears in the above equation, $\nabla V^{i,i'}_{s_i,s_{i'}}$ remains the same for any $i' \notin N_G(i)$.

In the case of $i' \in N_G(i)$, Jacobian has the form

$$\nabla V^{i,i'}_{s_i,s_{i'}} = \sum_{\bar{s}\in\bar{S}} u(s_i, s_{i'}, \bar{s}) Pr(\bar{s} \mid \bar{\sigma}) \qquad \text{where } Pr(\bar{s} \mid \bar{\sigma}) = \prod_{j\in N_G(i)-\{i,i'\}} \bar{\sigma}_j(\bar{s}_j) \tag{11}$$

Define $D = \max\{N_G(i), \forall i\}$, $\alpha$ as the maximal number of actions per agent. Then the computation of the Jacobian requires $O(|N|D\alpha^D + |N|^2)$ time. Notice that in many real settings of games, $D \ll N$. We can reduce the computation cost by an exponential factor $\alpha^{(N-D)}$.

## 4.2 Computing the Jacobian in action-graph games

Similar to agent graph games, the computation of the Jacobian of $V$ is the bottleneck of finding Nash equilibria of action-graph games. Recall the equation for computing the Jacobian of V in the general case, we show the equation of computing the Jacobian of V below for action graph game case.

$$\begin{aligned} \nabla V^{i,i'}_{s_i,s_{i'}} &= \sum_{\bar{s}^{(s_i)}\in\bar{S}^{(s_i)}} u(s_i, D(s_i, s^{(s_i)}_{i'}, \bar{s}^{(s_i)})) Pr(\bar{s}^{(s_i)} \mid \bar{\sigma}^{(s_i)}) \\ & \quad \text{where } Pr(\bar{s}^{(s_i)} \mid \bar{\sigma}^{(s_i)}) = \prod_{j\in\bar{N}} \bar{\sigma}^{(s_i)}_j(\bar{s}^{(s_i)}_j) \end{aligned} \tag{12}$$

Recall that the superscript in the equation denotes the projection of action $i$, which constructs a reduced graph by including only the nodes $v(s_i)$ and a new node $\phi$. We define the maximum indegree of the graph $G$ as $I$, so the maximum indegree in the projected graph $G^{(s)}$ is $I + 1$.

Because of the property represented in Equation (2), we treat $\bar{s}$ and $\bar{s}'$ as equivalent if and only if $D(\bar{s}) = D(\bar{s}')$. We only need to consider the projected distribution $\bar{D}^{(s_i)}$ rather than the projected pure action profile $\bar{s}^{(s_i)}$, obtaining

$$\begin{aligned} \nabla V^{i,i'}_{s_i,s_{i'}} &= \sum_{\bar{D}^{(s_i)}\in\bar{\Delta}^{(s_i)}} u(s_i, D(s_i, s^{(s_i)}_{i'}, \bar{D}^{(s_i)})) Pr(\bar{D}^{(s_i)} \mid \bar{\sigma}^{(s_i)}) \\ & \quad \text{where } Pr(\bar{D}^{(s_i)} \mid \bar{\sigma}^{(s_i)}) = \sum_{\bar{s}^{(s_i)}\in S(\bar{D}^{(s_i)})} Pr(\bar{s}^{(s_i)} \mid \bar{\sigma}^{(s_i)}) \end{aligned} \tag{13}$$

**Theorem 1** *Computation of the Jacobian for arbitrary action-graph games using Equation (13) takes time that is* $O((I+1)^{|\bar{N}|} poly(|\bar{N}|) poly(|S|))$.

Moreover, for a graphical game encoded as an AGG, by projecting the action graph, we obtain the same exponential speed up as in the graphical game case, which was illustrated in section 4.1.

In the symmetric case of action graph games, all agents have the same action choices: $\forall i, \forall j, S_i = S_j = S$. Nash proved that all finite symmetric games have at least one symmetric equilibrium. In this equilibrium, every agent uses the same mixed strategy. It is easy to show that finding the symmetric equilibrium is much easier than finding an arbitrary equilibrium. We add the same bonus to form a perturbed game. The initial equilibrium of the perturbed game is that all agents take the same action. In every step of continuation method, we always get a symmetric equilibrium as decreasing $\lambda$.

Since all agents have the same strategies, each pure action profile is equally likely. Hence for any $\bar{s} \in S(\bar{D}^{(s_i)})$

$$\begin{aligned} Pr(\bar{D}^{(s_i)} \mid \sigma_*^{(s_i)}) &= |S(\bar{D}^{(s_i)})| \, Pr(\bar{s}^{(s_i)} \mid \bar{\sigma}_*^{(s_i)}) \\ &\quad \text{where } Pr(\bar{s}^{(s_i)} \mid \bar{\sigma}_*^{(s_i)}) = \prod_{a \in \bar{S}^{(s_i)}} (\bar{\sigma}_j^{(s_i)}(a))^{\bar{D}^{(s_i)}(a)} \end{aligned} \tag{14}$$

Note that size $|S(\bar{D}^{(s_i)})|$ is the multinomial coefficient. The largest value of this size is reached when agents are evenly distributed across the nodes of the projected graph.

The computation of the Jacobin is much easier because we do not need to consider individual agent identities in $V$. We can use $\nabla V_{*\, s_i, s_{i'}}$ to replace $\nabla V^{i,i'}_{s_i, s_{i'}}$ for any $i \neq i'$. The Jacobian has the following form for symmetric case:

$$\nabla V_{*\, s_i, s_{i'}}(\sigma_*) = \sum_{\bar{D}^{(s_i)} \in \bar{\Delta}^{(s_i)}} u(s_i, D(s_i, s_{i'}^{(s_i)}, \bar{D}^{(s_i)})) Pr(\bar{D}^{(s_i)} \mid \bar{\sigma}_*^{(s_i)}) \tag{15}$$

Where $Pr(\bar{D}^{(s_i)} \mid \bar{\sigma}_*^{(s_i)})$ is defined in Equation (14).

**Theorem 2** *Computation of the Jacobian for symmetric action-graph games using Equation (15) takes time that is* $O(poly(|\bar{N}|^I) poly(|S|))$.

If readers are interested in the proofs about Theorem 1, 2, please refer the details in [3].

## 5 Conclusion

Since Nash proved that for every finite normal form game there exists at lease one Nash equilibrium, finding Nash equilibria became an active area of research. Even with the best known algorithm, computing a Nash equilibrium is exponential in computation. In the last few years, researchers have turned their focus to compact games. For graphical games, the exact algorithms are proposed but can only be applied to an undirected tree with every agent having only two actions. There are several algorithms introduced for computing $\epsilon$-Nash equilibria. Their running time either is bounded on the tree-width of the graph or has no bounds at all. Blum [5] shows that finding $\epsilon$-Nash equilibrium is not necessarily a step toward finding an exact equilibrium. In fact, $\sigma$ being an $\epsilon$-Nash equilibrium does not guarantee the existence of an exact equilibrium in the neighborhood of $\sigma$.

In this paper, we introduced two types of most important compact game representations, agent graph game and action graph game. Those two game models explore the topological relationship between nodes in the game graph in different respect. Agent graph game considers each agent as a node in the game graph and presents the dependency of payoff between agents by edges. If there exists payoff dependency between agent $i$ and $j$, then there exists an edge between $i$ and $j$. For action graph games, actions are represented as nodes in the game graph, and the edges between two nodes represent the payoff dependency of the corresponding actions of agents. AGG can be used to encode both strict and context-specific independence.

The continuation method is the most recent advanced method to compute Nash equilibria. The bottleneck of computing equilibria using the continuation method is computing the Jacobian of the payoff function. We show that exponential savings can be achieved by applying the continuation method to compact games. In the graphic game case, the computation cost of the Jacobian is only exponential in the graphical game's maximal number of node's degree instead of total number of nodes. Note that in most cases, the maximal degree of nodes are much less than the total number of nodes in the graph. For the action graph, we show that computation of the Jacobian grows exponentially with the action graph's maximum degree of the nodes rather than with its total number of nodes for an arbitrary case. In an important case of AGG, symmetric game, the Jacobian can be computed in polynomial time when the action's maximal in-degree is constant.

## References

[1] Shoham, Y., & Leyton-Brown, K. (2005). Multi Agent Systems (draft book).

[2] Kearns, M., Littman, M., & Singh, S. (2001). Graphical models for game theory. *UAI*.

[3] Bhat, N., & Leyton-Brown, K. (2004). Computing Nash equilibria of action-graph games. *UAI*.

[4] Vickrey, D., & Koller, D. (2002). Multi-agent algorithms for solving graphical games. *AAAI*.

[5] Blum, B., Shelton, C., & Koller, D. (2003). A continuation method for Nash equilibria in structured games. *IJCAI*.

[6] Govindan, S., & Wilson, R. (2003). A global Newton method to compute Nash equilibria. *J. Economic Theory, 110*, 65-86.