A Quantum Annealing Algorithm for Finding Pure Nash Equilibria in Graphical Games

Christoph Roch, Thomy Phan, Sebastian Feld, Robert Müller, Thomas Gabor and Claudia Linnhoff-Popien
LMU Munich
christoph.roch@ifi.lmu.de

Abstract

We introduce Q-Nash, a quantum annealing algorithm for the NP-complete problem of finding pure Nash equilibria in graphical games. The algorithm consists of two phases. The first phase determines all combinations of best response strategies for each player using classical computation. The second phase finds pure Nash equilibria using a quantum annealing device by mapping the computed combinations to a quadratic unconstrained binary optimization formulation based on the Set Cover problem. We empirically evaluate Q-Nash on D-Wave’s Quantum Annealer 2000Q using different graphical game topologies. The results with respect to solution quality and computing time are compared to a Brute Force algorithm and a Random Search heuristic.

1 Introduction

Applications of conventional game theory have played an important role in many modern strategic decision making processes including diplomacy, economics, national security, and business [Carfi et al., 2011; Rabin, 1993; Roy et al., 2010]. Game theory is a mathematical paradigm in which such domain-specific decision situations are modeled [Fudenberg and Tirole, 1991]. Multiple players interact with each other to collectively complete a task or to enforce their interests. In the classical model of game theory [Von Neumann and Morgenstern, 1944] all players choose an action simultaneously and obtain a certain payoff (utility), which depends on the actions of the other players. The most common solution concept for such a decision problem is called Nash equilibrium (NE) [Nash, 1951], in which no player is able to unilaterally improve his payoff by changing his chosen action.

There exist many different representations for such simultaneous games. The most popular is the strategic or standard normal-form game representation, which is often used for 2-player games, like the prisoner dilemma or battle of the sexes. However, due to its exponential growth of the representational size w.r.t. the number of players [Kearns et al., 2013], a more compact version, called graphical game, is increasingly used to model multi-player scenarios [La Mura, 2000; Koller and Milch, 2003; Littman et al., 2002; Palmieri and Lallouet, 2017]. Here a player’s action only depends on a certain number of other players’ actions (a so called player’s neighborhood). These neighborhoods are visualized by an underlying graph with players as vertices and the dependencies as edges.

While pure strategies, where each player unambiguously decides on a particular action, are conceptually simpler than mixed strategies, the associated computational problems appear to be harder [Gottlob et al., 2005]. This also applies to the compact representation of graphical games, for which the complexity of finding pure strategy Nash equilibria (PNE-GG) was proven to be NP-complete, even in the restricted case of neighborhoods of maximum size 3 with at most 3 actions per player [Gottlob et al., 2005].

Although there are many algorithms that find mixed or approximated NE in graphical games [Kearns et al., 2013; Ortiz and Kearns, 2003; Bhat and Leyton-Brown, 2004; Soni et al., 2007], there are only a couple of algorithms that deal with pure Nash equilibria (PNE). In this paper, we focus on finding the hard computational Nash equilibria in the case of pure strategies, where each player chooses to play an action in a deterministic, non-random manner.

With D-Wave Systems releasing the first commercially available quantum annealer in 2011\footnote{https://www.dwavesys.com/news/d-wave-systems-sells-its-first-quantum-computing-system-lockheed-martin-corporation}, there is now the possibility to find solutions for such complex problems in a completely different way compared to classical computation. To use D-Wave’s quantum annealer, the problem has to be formulated as a quadratic unconstrained binary optimization (QUBO) problem [Boros et al., 2007], which is one possible input type for the annealer. In doing so, the metaheuristic quantum annealing seeks to find the minimum of an objective function, i.e., the best solution of the defined configuration space [McGeoch, 2014].

In this paper, we propose the first quantum annealing algorithm for finding PNE-GG, called Q-Nash. The algorithm consists of two phases. The first phase determines all combinations of best response strategies for each player using classical computation. The second phase finds pure Nash Equilibria using a quantum annealing device by mapping the computed combinations to a QUBO formulation based on the Set Cover problem. We empirically evaluate Q-Nash on D-
Wave’s Quantum Annealer 2000Q using different graphical game topologies. The results with respect to solution quality and computing time are compared to a Brute Force algorithm and a Random Search heuristic.

2 Background

2.1 Graphical Games and Pure Nash Equilibria

In an \( n \)-player game, each player \( p \) (\( 1 \leq p \leq n \)), has a finite set of strategies or actions, \( S_p \), with \( |S_p| \geq 2 \). Such a game can be visualized by a set of \( n \) matrices \( M_p \). The entry \( M_p(s_1, \ldots, s_n) = M_p(s) \) specifies the payoff to player \( p \) when the joint action (also, strategy profile) of the \( n \) players is \( s \in S \), with \( S = \prod_{i=1}^{n} S_i \) being the set of combined strategy profiles. In order to specify a game with \( n \) players and \( s \) strategies each, the representational size is \( n^s \), an amount of information exponential with respect to the number of players. However, players often interact only with a limited number of other players, which allows for a much more succinct representation. In [Kearns et al., 2013] such a compact representation, called graphical game, is defined as follows:

**Definition 2.1. (Graphical Game)**

An \( n \)-player graphical game is a pair \( (G, M) \), where \( G \) is an undirected graph with \( n \) vertices and \( M \) is a set of \( n \) matrices \( M_p \) with \( 1 \leq p \leq n \), called the local game matrices. Player \( p \) is represented by a vertex labeled \( p \) in \( G \). We use \( N_G(p) \subseteq \{1, \ldots, n\} \) to denote the set of neighbors of player \( p \) in \( G \). By convention, \( N_G(p) \) always includes \( p \) itself. The interpretation is that each player is in a game with only his neighbors in \( G \). Thus, the size of the graphical game representation is only exponential in the maximal node degree \( \delta \) of the graph, \( n\delta^d \). If \( |N_G(p)| = k \) and \( s \in \prod_{i=1}^{n} S_i \), \( M_p(s) \) denotes the payoff to \( p \) when his \( k \) neighbors (including himself) play \( s \).

Consider a game with \( n \) players and strategy sets \( S_1, \ldots, S_n \). For every strategy profile \( s \in S \), the strategy of player \( p \) is denoted by \( s_p \) and \( s_{-p} \) corresponds to the \( (n-1) \)-tuple of strategies of all players but \( p \). For every \( s_p \in S_p \) and \( s_{-p} \in S_{-p} \), we denote by \( (s_p; s_{-p}) \) the strategy profile in which player \( p \) plays \( s_p \), and all the other players play according to \( s_{-p} \). One has to mention that a strategy profile \( s \) is called global, if all \( n \) players contribute to it, i.e., a global combined strategy \( s \) consists of every player playing one of his actions [Gottlob et al., 2005].

**Definition 2.2. (Pure Nash Equilibrium)**

A global strategy profile \( s \) is a PNE, if for every player \( p \) and strategy \( s_p \in S_p \) we have \( M_p(s) \geq M_p(s_{-p}; s_p) \). That is, no player can improve his expected payoff by deviating unilaterally from a Nash Equilibrium.

[Daskalakis and Papadimitriou, 2006] define a best response strategy as follows:

**Definition 2.3. (Best Response Strategy)**

A best response strategy of player \( p \) is defined by:

\[
BR_{M_p}(s_{-p}) \triangleq \{s_p \mid s_p \in S_p \text{ and } \forall s_p' \in S_p : M_p(s_{-p}; s_p') \geq M_p(s_{-p}; s_p') \}
\]

Intuitively, \( BR_{M_p}(s_{-p}) \) is the set of strategies in \( S_p \) that maximize \( p \)'s payoff if the other players play according to \( s_{-p} \). Thus, a strategy profile \( s \) is a pure Nash equilibrium if for every player \( p \), \( s_p \in BR_{M_p}(s_{-p}) \).

2.2 Quantum Annealing

**Quantum annealing** is a metaheuristic for solving complex optimization and decision problems [Kadowaki and Nishimori, 1998]. D-Wave’s quantum annealing heuristic is implemented in hardware, designed to find the lowest energy state of a spin glass system, described by an Ising Hamiltonian,

\[
H(s) = \sum_i h_i x_i + \sum_{i<j} J_{ij} x_i x_j
\]

where \( h_i \) is the on-site energy of qubit \( i \), \( J_{ij} \) are the interaction energies of two qubits \( i \) and \( j \), and \( x_i \) represents the spin \((-1, +1)\) of the qubit. The basic process of quantum annealing is to physically interpolate between an initial Hamiltonian \( H_I \) and an easy to prepare minimal energy configuration (or ground state), and a problem Hamiltonian \( H_P \), whose minimal energy configuration is sought that corresponds to the best solution of the defined problem (see Eq. 2). This transition is described by an adiabatic evolution path which is mathematically represented as function \( s(t) \) and decreases from 1 to 0 [McGeoch, 2014].

\[
H(t) = s(t)H_I + (1 - s(t))H_P
\]

If this transition is executed sufficiently slow, the probability to find the ground state of the problem Hamiltonian is close to 1 [Albash and Lidar, 2018]. Thus, by mapping the Nash equilibrium decision problem onto a spin glass system, quantum annealing is able to find the solution of it.

For completeness, we map our NE decision problem to an alternative formulation of the Ising spin glass system. The so-called QUBO problem [Boros et al., 2007] is mathematically equivalent and uses 0 and 1 for the spin variables [Su et al., 2016]. The quantum annealer is as well designed to minimize the functional form of the QUBO:

\[
\min x^T Q x \text{ with } x \in \{0, 1\}^n
\]

with \( x \) being a vector of binary variables of size \( n \), and \( Q \) being an \( n \times n \) real-valued matrix describing the relationship between the variables. Given the matrix \( Q : n \times n \), the annealing process tries to find binary variable assignments \( x \in \{0, 1\}^n \) to minimize the objective function in Eq. 3.

2.3 Set Cover Problem

Since the QUBO formulation of Q-Nash resembles the well known Set Cover (SC) problem, it is introduced here. Within the SC problem, one has to find the smallest possible number of subsets from a given collection of subsets \( V_k \subseteq U \) with \( 1 \leq k \leq N \), such that the union of them is equal to a global superset \( U \) of size \( n \). This problem was proven to be NP-hard [Karp, 1972]. In [Lucas, 2014] the QUBO formulation for the Set Cover problem is given by:

\[
H = A \sum_{\alpha=1}^{n} \left( 1 - \sum_{m=1}^{N} x_{\alpha,m} \right)^2 + A \sum_{\alpha=1}^{n} \left( \sum_{m=1}^{N} \alpha x_{\alpha,m} - \sum_{k=1}^{N} V_k \right)^2
\]
with \( x_k \) being a binary variable which is 1, if set \( k \) is included within the chosen sets, and 0 otherwise. \( x_{\alpha,m} \) denotes a binary variable which is 1 if the number of chosen subsets \( V_k \) which include element \( \alpha \) is \( m \geq 1 \), and 0 otherwise. The first energy term imposes the constraints that for any given \( \alpha \) exactly one \( x_{\alpha,m} \) must be 1, since each element of \( U \) must be included a fixed number of times, and that the number of times that we claimed \( \alpha \) was included is in fact equal to the number of subsets \( V_k \) we have included with \( \alpha \) as an element. \( A \) is a penalty value, which is added on top of the solution energy, described by \( H \), if a constraint was not satisfied, i.e. one of the two terms (quadratic differences) are unequal to 0. Therefore adding a penalty value states a solution as invalid. Additionally, the SC problem minimizes over the number of chosen subsets \( V_k \). We skip discussing the term due to the fact that it has no impact on our Q-Nash QUBO problem later on.

### 3 Related Work

Most known algorithms focus on finding mixed or approximated NE [Kearns et al., 2013; Ortiz and Kearns, 2003; Bhat and Leyton-Brown, 2004; Soni et al., 2007]. However, some investigations in determining PNE in graphical and similar variations of games were made.

Daskalakis and Papadimitriou present a reduction from graphical games to Markov random fields such that PNE can be found by statistical inference. They use known statistical inference algorithms like belief propagation, junction tree algorithm and Markov Chain Monte Carlo to determine PNE in graphical games [Daskalakis and Papadimitriou, 2006].

In Jiang and Leyton-Brown, 2007], the authors analyze the problem of computing pure Nash equilibria in action graph games (AGGs), another compact game theoretic representation, which is similar to graphical games. They propose a dynamic-programming approach that constructs equilibria of the game from equilibria of restricted games played on subgraphs of the action graph. In particular, under the premise that the game is symmetric and the action graph has bounded treewidth, their algorithm determines the existence of a PNE in polynomial time.

Palmieri and Lallouet deal with constraint games, for which constraint programming is used to express players preferences. They rethink their solving technique in terms of constraint propagation by considering players preferences as global constraints. Their approach is able to find all pure Nash equilibria for some problems with 200 players and also shows that performance can be improved for graphical games [Palmieri and Lallouet, 2017].

With quantum computing gaining more and more attention and none of the related work making use of quantum annealing in order to find PNE, we propose a solution approach using a quantum annealer.

### 4 Q-Nash

In the following sections we present the concept of Q-Nash. Q-Nash consists of two phases, which are described below.

---

#### 4.1 Determining best response strategies

In the first phase, we identify each player’s best response to what the other players might do. That is, for every strategy profile \( s \), we search player \( p \)'s strategy (or strategies) with the maximum payoff \( M_p(s) \). This involves iterating through each player in turn and determining their optimal strategies. An example is given in 4.1. This can be feasibly done in polynomial time, since one can easily explore each player’s matrix \( M_p \) representing the utility function, i.e., payoffs [Gottlob et al., 2005]. Therefore the first part, which we see as preliminary step for our Q-Nash algorithm, is executed on a classical computer. After doing this, one gets a set of combined strategies with each being a best response to the other players’ played strategies. This set is denoted by \( B = \{BR_{M_p} \mid 1 \leq p \leq n \} \) and has the cardinality \( C_B = \sum_{p=1}^{n} |BR_{M_p}| \).

**Example 4.1.** In Tab. 1 the local payoff matrix \( M \) of player \( A \) is visualized. For instance, assume player \( B \) plays action 2 and player \( C \) chooses action 1. In this case, a best response strategy for player \( A \) is action 0, due to the fact, that he gets the most payoff in this situation, i.e. 4. This leads to a best response strategy combination for player \( A \), denoted by a pointed set \( \{A0, B2, C1\} \) with player \( A \) being the base point of it.

|     | B0 C0 | B0 C1 | B1 C0 | B1 C1 | B2 C0 | B2 C1 |
|-----|-------|-------|-------|-------|-------|-------|
| 0   | 4     | 1     | 2     | 2     | 1     | 4     |
| 1   | 1     | 3     | 2     | 1     | 2     | 2     |

Table 1: Local dependency payoff matrix \( M \) of player \( A \). The payoffs marked in bold correspond to a best response strategy of player \( A \).

#### 4.2 Finding PNE using quantum annealing

In the second phase, a PNE is identified, when all players are playing one of their best response strategies simultaneously. With the computed set \( B \) of the classical phase, the following question arises:

"Is there a union of the combined best response strategies of \( B \) which results in a global strategy profile, under the premise that every player plays one of his best response actions?" — As stated in Definition 2.3 this would lead to a PNE.

This question resembles the Set Cover (SC) problem, stated in Sec. 2.3. It asks for the smallest possible number of subsets to cover the elements of a given global set (in our case, this global set would be a feasible global strategy profile and the subsets correspond to our best response strategy profiles of \( B \)). However, for our purposes we have to modify the given formulation in Eq. 4 as follows:

\[
H = A \sum_{p=1}^{n} \left( 1 - \sum_{j=1}^{n} \sum_{m=1}^{\infty} x_{p,j,m} \right)^2 + A \sum_{p=1}^{n} \sum_{j=1}^{n} \sum_{m=1}^{\infty} m x_{p,j,m} \sum_{k,j \neq m} x_{k} + A \left( \sum_{k=1}^{n} x_{k} \right)^2
\]

---

3https://www.gartner.com/smarterwithgartner/the-cios-guide-to-quantum-computing/
Eq. 5 is quite similar to Eq. 4. However, an element \( \alpha \) of the superset of Eq. 4 corresponds to a player \( p \) and his chosen action \( j \) of our global strategy profile. Nevertheless, the intention of those first two energy terms complies with the intention of Eq. 4, stated in Sec. 2.3. Further, another energy term must be added to our QUBO problem as constraint. This last energy term, for which an instance is given in Example 4.2, states that exactly \( n \) sets of \( B \) should be included to form the global strategy profile. This constraint implicitly ensures that every player is playing one of his best response strategies.

\( \alpha \) is called penalty value, which is added on top of the solution energy, if a constraint was not satisfied, i.e. one of the three terms (quadratic differences) are unequal to 0. Thus, adding a penalty value states a solution as invalid. Only if the total energy described by \( H = 0 \) the corresponding solution is a valid global best response strategy profile and thus a PNE.

All these energy terms are specified within the QUBO problem matrix \( Q \), which the quantum annealer takes as an input, goes through the annealing process and responds with a binary vector \( x \) as a solution, see Sec. 2.2. This vector indicates which best response strategy of each player should be chosen to form a PNE.

**Example 4.2.** To demonstrate the function of the last energy term (constraint) of Eq. 5, an excerpt of best response strategy combinations of \( B \) (in form of pointed sets, with the bold player-action-combination being the base point) for an arbitrary 4-player game are visualized in Fig. 1. The green union of four best response sets leads to a PNE, in which every player is playing one of his best response strategies. Although the red union of three sets also leads to a global combined strategy set, in which every player is playing one of his actions, it is not a PNE, due to player \( B \) not playing a best response strategy.

![Figure 1](https://example.com/figure1.png)

**Figure 1:** On the left hand an excerpt of superset \( B \) is given. On the right hand the sets are united, with each (1&2) being a global strategy combination, but only one (1) being a PNE.

### 5 Experiments

#### 5.1 Evaluation Graph-Topologies

For evaluating Q-Nash, we implemented a game generator. It creates graphical game instances of three different popular graphical structures, which were often used in literature (Koller and Milch, 2003; Vickrey and Koller, 2002; Jiang and Leyton-Brown, 2007; Palmieri and Lallouet, 2017). These graphical structures are shown in Fig. 2. As an input for our game generator, one can choose the number of players, the graph-topology and thus the dependencies between the players and the number of actions for each player individually. The corresponding payoffs are sampled randomly from \([0, 15]\). For our experiments we considered games with three actions per player. The classic theorem of Nash [Nash, 1951] states that for any game, there exists a Nash equilibrium in the space of joint mixed strategies. However, in this work we only consider pure strategies and therefore there might be (graphical) games without any PNE (see, for instance, [Osborne and Rubinstein, 1994]). Additionally we want to emphasize, that Q-Nash is able to work on every graphical game structure, even if the dependency graph is not connected, for instance a set of trees (called forest).

![Figure 2](https://example.com/figure2.png)

**Figure 2:** Different graphical game structures (topologies) which indicate the dependencies between players of a game. (a) Tree-Topology with \( n = 10 \), (b) Circle-Topology with \( n = 6 \) and (c) Road-Topology with \( n = 6 \).

#### 5.2 Methods

**QBSolv**

Due to the fact, that quantum computing is still in its infancy, and corresponding hardware is limited in the number of qubits and their connectivity, we need to fall back to a hybrid method (QBSolv\(^3\)), in order to solve large problem instances. QBSolv is a software that automatically splits instances up into subproblems submitted to D-Wave’s quantum annealer, and an extensive tabu search is applied to post-process all D-Wave solutions. Additionally, QBSolv embeds the QUBO problem to the quantum annealing hardware chip. QBSolv further allows to specify certain parameters such as the number of individual solution attempts (num_repeats), the subproblem size used to split up instances which do not fit completely onto the D-Wave hardware and many more. For detailed information, see [Booth et al., 2017].

**Brute Force**

For evaluating the effectiveness of Q-Nash we implemented a Brute Force (BF) algorithm to compare with. It determines the best response strategy sets of all players in the same way as Q-Nash does and afterwards tries out every possible combination of those sets to form a valid global strategy set which corresponds to a PNE. The number of combinations is exponential with respect to the number of players of the game, \( \prod_{p=1}^{n} BR_{M_p} \).

**Random Search**

Additionally, a Random Search (RS) algorithm was implemented, which acts like the BF algorithm, but tries to ran-

---

\(^3\)https://github.com/dwavesystems/qbsolv
domly find combinations of best response sets, which correspond to a Nash equilibrium. Furthermore it takes a timespan as an input parameter and terminates after a timeout occurred.

5.3 Computational times of Q-Nash

With respect to the computational results stated in Tab. 2 we first introduce the time components of Q-Nash’s total computation time. For a better understanding a general overview of Q-Nash is given in Algorithm 1.

The *find embedding time* (3) states the time D-Wave’s classical embedding heuristic takes to find a valid subproblem hardware embedding. Determining best responses (4) describes the time Q-Nash classical computing phase takes to identify the best responses strategies of every player and additionally build up the QUBO matrix. The *QBSolv time* can be divided into the classical time (5) and the quantum annealing time (6). The classical *QBSolv time* (5) comprises of not only a tabu search, which iteratively processes all subproblem solutions, it also contains the latency and job queuing time to D-Wave’s quantum hardware. The *quantum annealing time* (6) comprises of the number of subproblems times D-Wave’s *qpu-access-time*.

Algorithm 1 General structure of Q-Nash

Input: Game instance as file
Parameter: List of QBSolv parameters (num_repeats, subproblem_size,...)
Methods:
1. embedding = find_hardware_embedding_for_subproblem_size(subproblem_size) (3)
2. Q = determine_best_response_strategies_and_build_qubo_matrix() (4)
3. binary_vector = run_QBSolv(Q, embedding, num_repeats, ...) (5)(6)
Output: All found PNE

6 Results & Discussion

We investigated the solution quality and computational time of Q-Nash. Fig. 3 shows the ability of finding PNE of the three proposed methods (Q-Nash with QBSolv, BF and RS) in differently structured graphical games. For every graph topology (Tree, Circle and Road) we used games with players ranging from 6 to 10, due to the fact that BF took too long to solve larger games. The results show that Q-Nash found the same amount of PNE as the exact BF algorithm for those instances. We ran RS as long as Q-Nash (total time) took to solve the instances. One can see that RS was only able to find one or two PNE in the smallest game instances with 6 players.

Because of Q-Nash performing well on these instances we tested it for larger game instances, see Fig. 4 and 5. For the Road- and Circle-Topology, the solution quality of Q-Nash was evaluated with different numbers of players ranging from 20 to 40. Running Q-Nash 20 times on every instance, one can see that Q-Nash was not always able to find the same amount of PNE per run. Although it seems that for larger scaled game instances the variance of finding PNE decreases, one has to take into account that these game instances might contain less PNE, due to the fact that all the players have to play one of their best response strategies to form a PNE. Furthermore, RS never found a PNE on those game instances, which highlights the huge combinatorial action space on which Q-Nash performs comparatively well with respect to the effectiveness.

Because of quantum annealing being a heuristic, one can set the number of annealing attempts with the *num_repeats* parameter of QBSolv, to improve the accuracy. In Fig. 6 the influence of this parameter is shown. We exemplary used a 25 player road game and ran Q-Nash 20 times per parameter setting to show its impact on the effectiveness. As expected one can see, that with increasing number of annealing attempts the inter-quartile range and its median in regard to the number of PNE found, increase. Although an annealing process takes only 20µs, it adds up with the *num_repeats* parameter and therefore leads to a trade-off between computational time and accuracy.

The computational results are for circle structured graphical games and can be viewed in Tab. 2. The used game instances differ in the number of players (6-20 Player) and the computing times are given in seconds for both, the BF and our Q-Nash algorithm. As already mentioned in Sec. 5.2 we need to use QBSolv to solve large problem instances and therefore we have to consider the computational results of Q-Nash with caution, as stated in Sec. 5.3.

Regarding the BF algorithm, the *total time* (1) is only given
for graphical games with maximal 10 players, since the computational time is exponential with respect to the number of players. For the 10 player game BF already took more than 1 day and 5 hours on an Intel Core i5-4590 CPU with 3.30GHz. However, due to Q-Nash’s immense overhead time (embedding, tabu search, latency, queuing time, etc.) the total time (2) is outperformed by BF in relation to the 6 player game. Nevertheless the pure Q-Nash computational time (determining best response strategies (4) & QA time (6)) is comparatively very small (1.359 sec).

In general, one can see that the quantum annealing time (6) for these game instances is quite constant and the runtime of the classical Q-Nash phase (4) is only polynomial, as stated in Section 4.1.

With the embedding time (3) being constant, the only unpredictable runtime component of Q-Nash is QBSolv classic (5). This is due to the tabu search, which only terminates after a specified number of iterations in which no improvement of the previously found solutions could be made.

However, with increasing number of qubits and their connectivity it might be possible to map larger game instances directly to the hardware chip such that the hybrid solver QBSolv can be avoided. Thus, the total Q-Nash time would only consist of the classical algorithm phase time (4), the embedding time (3) and a fraction of the quantum annealing time (6), since we do not have to split our problem instance into subproblems anymore.

### 7 Conclusion

We proposed Q-Nash, to our knowledge the first algorithm that finds PNE-GG using quantum annealing hardware.

Regarding the effectiveness of Q-Nash, we showed that for small game instances (ranging from 6-10 players) the algorithm was always able to find all PNE in differently structured graphical games. Anyway we have to mention, that with increasing number of players the variance w.r.t the number of found PNE increased, too.

Due to the fact that quantum computing is still in its infancy and recent hardware is limited in the number of qubits and their connectivity, we had to fall back on a quantum-classical hybrid solver, called QBSolv, which involves additional overhead time. That makes it difficult to draw a fair comparison of Q-Nash and classical state-of-the-art solution methods regarding the computational time. We therefore decomposed the total time into its main components to show their impact. According to the experimental results, the only unpredictable time component is QBSolv’s classical tabu search along with latency and job queuing times at D-Wave’s cloud computing frontend. However with D-Wave announcing an immense rise of the number of qubits and their connectivity on D-Wave’s quantum processors in the next years\(^4\), it might be possible to embed larger game instances directly onto the chip and therefore omit hybrid solvers like QBSolv. Another possibility is using Fujitsu’s Digital Annealing Unit (DAU) which also takes a QUBO matrix as input. With DAU being able to solve larger fully connected QUBO problems [Aramon et al., 2019], a shorter total computation time could be achieved.

\(^4\)https://www.dwavesys.com/sites/default/files/mwj_dwave_qubits2018.pdf

### Table 2: Computational times of the Q-Nash and Brute Force (BF) algorithms on various circle structured graphical games

|       | 6 Player | 8 Player | 10 Player | 12 Player | 14 Player | 16 Player | 18 Player | 20 Player |
|-------|----------|----------|-----------|-----------|-----------|-----------|-----------|-----------|
| BF    |          |          |           |           |           |           |           |           |
| (1) Total time [s] | 6.235 | 428.344 | 107486.286 | - | - | - | - | - |
| (2) Total time [s] | 85.393 | 125.828 | 457.906 | 424.701 | 273.283 | 228.609 | 276.593 | 248.547 |
| Q-Nash |          |          |           |           |           |           |           |           |
| (3) Find embedding (Classic) [s] | 4.172 | 4.062 | 4.093 | 5.155 | 4.813 | 5.703 | 4.202 | 4.391 |
| (4) Determining best responses (Classic) [s] | 0.391 | 0.891 | 1.828 | 3.063 | 4.906 | 7.328 | 10.704 | 14.360 |
| (5) QBSolv (Classic) [s] | 79.862 | 120.012 | 448.834 | 412.869 | 260.285 | 212.849 | 258.395 | 226.775 |
| (6) QA time (Quantum) [s] | 0.968 | 0.863 | 3.151 | 3.614 | 3.279 | 2.729 | 3.292 | 3.021 |

### Figure 5: Variance of found PNE – Q-Nash on Circle-Topology games with different numbers of players.

### Figure 6: The impact of num_repeats parameter on a 25 player road game.

## References

1. Aramon, et al. (2019). Quantum computing for graph problems: Quantum annealing on fully connected problems. *Physical Review A*, 100(6), 062324.
2. D-Wave (2018). D-Wave 2000 QPU. Retrieved from https://www.dwavesys.com/sites/default/files/mwj_dwave_qubits2018.pdf

---

For graphical games with maximal 10 players, since the computational time is exponential with respect to the number of players. For the 10 player game BF already took more than 1 day and 5 hours on an Intel Core i5-4590 CPU with 3.30GHz. However, due to Q-Nash’s immense overhead time (embedding, tabu search, latency, queuing time, etc.) the total time (2) is outperformed by BF in relation to the 6 player game. Nevertheless the pure Q-Nash computational time (determining best response strategies (4) & QA time (6)) is comparatively very small (1.359 sec).

In general, one can see that the quantum annealing time (6) for these game instances is quite constant and the runtime of the classical Q-Nash phase (4) is only polynomial, as stated in Section 4.1.

With the embedding time (3) being constant, the only unpredictable runtime component of Q-Nash is QBSolv classic (5). This is due to the tabu search, which only terminates after a specified number of iterations in which no improvement of the previously found solutions could be made.

However, with increasing number of qubits and their connectivity it might be possible to map larger game instances directly to the hardware chip such that the hybrid solver QBSolv can be avoided. Thus, the total Q-Nash time would only consist of the classical algorithm phase time (4), the embedding time (3) and a fraction of the quantum annealing time (6), since we do not have to split our problem instance into subproblems anymore.

### 7 Conclusion

We proposed Q-Nash, to our knowledge the first algorithm that finds PNE-GG using quantum annealing hardware.

Regarding the effectiveness of Q-Nash, we showed that for small game instances (ranging from 6-10 players) the algorithm was always able to find all PNE in differently structured graphical games. Anyway we have to mention, that with increasing number of players the variance w.r.t the number of found PNE increased, too.

Due to the fact that quantum computing is still in its infancy and recent hardware is limited in the number of qubits and their connectivity, we had to fall back on a quantum-classical hybrid solver, called QBSolv, which involves additional overhead time. That makes it difficult to draw a fair comparison of Q-Nash and classical state-of-the-art solution methods regarding the computational time. We therefore decomposed the total time into its main components to show their impact. According to the experimental results, the only unpredictable time component is QBSolv’s classical tabu search along with latency and job queuing times at D-Wave’s cloud computing frontend. However with D-Wave announcing an immense rise of the number of qubits and their connectivity on D-Wave’s quantum processors in the next years\(^4\), it might be possible to embed larger game instances directly onto the chip and therefore omit hybrid solvers like QBSolv. Another possibility is using Fujitsu’s Digital Annealing Unit (DAU) which also takes a QUBO matrix as input. With DAU being able to solve larger fully connected QUBO problems [Aramon et al., 2019], a shorter total computation time could be achieved.

\(^4\)https://www.dwavesys.com/sites/default/files/mwj_dwave_qubits2018.pdf
References

[Aramon et al., 2019] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut Katzgraber. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Bulletin of the American Physical Society*, 2019.

[Bhat and Leyton-Brown, 2004] Navin AR Bhat and Kevin Leyton-Brown. Computing nash equilibria of action-graph games. In *Proc. of the 20th conference on Uncertainty in artificial intelligence*, pages 35–42. AUAI Press, 2004.

[Booth et al., 2017] Michael Booth, Steven P Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical. *quantum execution*. Technical Report, pages 01–09, 2017.

[Boros et al., 2007] Endre Boros, Peter L Hammer, and Gabriel Tavares. Local search heuristics for quadratic unconstrained binary optimization (qubo). *Journal of Heuristics*, 13(2):99–132, 2007.

[Carfì et al., 2011] David Carfì, Francesco Musolino, et al. Fair redistribution in financial markets: a game theory complete analysis. *Journal of Advanced Studies in Finance*, 2(2):4, 2011.

[Daskalakis and Papadimitriou, 2006] Constantinos Daskalakis and Christos H Papadimitriou. Computing pure nash equilibria in graphical games via markov random fields. In *Proc. of the 7th ACM conference on Electronic commerce*, pages 91–99. ACM, 2006.

[Fudenberg and Tirole, 1991] Drew Fudenberg and Jean Tirole. Game theory. 1991. *Cambridge, Massachusetts*, 393(12):80, 1991.

[Gottlob et al., 2005] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research*, 24:357–406, 2005.

[Jiang and Leyton-Brown, 2007] Albert Xin Jiang and Kevin Leyton-Brown. Computing pure nash equilibria in symmetric action graph games. In *AAAI*, volume 1, pages 79–85, 2007.

[Kadowaki and Nishimori, 1998] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58(5):5355, 1998.

[Karp, 1972] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[Kearns et al., 2013] Michael Kearns, Michael L Littman, and Satinder Singh. Graphical models for game theory. *arXiv preprint arXiv:1301.2281*, 2013.

[Koller and Milch, 2003] Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and economic behavior*, 45(1):181–221, 2003.

[La Mura, 2000] Pierfrancesco La Mura. Game networks. In *Proc. of the 16th conference on Uncertainty in artificial intelligence*, pages 335–342, 2000.

[Littman et al., 2002] Michael L Littman, Michael J Kearns, and Satinder P Singh. An efficient, exact algorithm for solving tree-structured graphical games. In *Advances in Neural Information Processing Systems*, pages 817–823, 2002.

[Lucas, 2014] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2:5, 2014.

[McGeoch, 2014] Catherine C McGeoch. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing*, 5(2):1–93, 2014.

[Nash, 1951] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[Ortiz and Kearns, 2003] Luis E Ortiz and Michael Kearns. Nash propagation for loopy graphical games. In *Advances in Neural Information Processing Systems*, pages 817–824, 2003.

[Osborne and Rubinstein, 1994] Martin J Osborne and Ariel Rubinstein. A course in game theory. MIT press, 1994.

[Palmieri and Lallouet, 2017] Anthony Palmieri and Arnaud Lallouet. Constraint games revisited. In *International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 729–735, 2017.

[Rabin, 1993] Matthew Rabin. Incorporating fairness into game theory and economics. *The American economic review*, pages 1281–1302, 1993.

[Roy et al., 2010] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10. IEEE, 2010.

[Soni et al., 2007] Vishal Soni, Satinder Singh, and Michael P Wellman. Constraint satisfaction algorithms for graphical games. In *Proc. of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 61, 2007.

[Su et al., 2016] Juexiao Su, Tianheng Tu, and Lei He. A quantum annealing approach for boolean satisfiability problem. In *Proc. of the 53rd Annual Design Automation Conference*, page 148. ACM, 2016.

[Vickrey and Koller, 2002] David Vickrey and Daphne Koller. Multi-agent algorithms for solving graphical games. In *AAAI/IAAI*, pages 345–351, 2002.

[Von Neumann and Morgenstern, 1944] John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior. 1944.