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

Christoph Roch(✉), Thomy Phan, Sebastian Feld, Robert Müller, Thomas Gabor, Carsten Hahn, and Claudia Linnhoff-Popien

LMU Munich, Munich, Germany
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 the Iterated Best Response heuristic.

Keywords: Quantum annealing · Game theory · Optimization

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 [6,24,26]. Game theory is a mathematical paradigm in which such domain-specific decision situations are modeled [9]. Multiple players interact with each other to collectively complete a task or to enforce their interests. In the classical model of game theory [30] 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)* [20], 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 [14], a more compact version, called graphical game,

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12142, pp. 488–501, 2020.
https://doi.org/10.1007/978-3-030-50433-5_38

is increasingly used to model multi-player scenarios [15–17,23]. 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 [10]. 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 [10].

Although there are many algorithms that find *mixed* or *approximated* NE in graphical games [3,14,21,27], there are only a couple of algorithms that deal with *pure Nash equilibria (PNE)*. In this paper, we focus on the computationally hard problem of finding NE with *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[1], 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 [5], 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 [19].

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 the Iterated Best Response 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 $M_p(s_1, ..., s_n) = M_p(s)$ 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

[1] https://www.dwavesys.com/news/d-wave-systems-sells-its-first-quantum-computing-system-lockheed-martin-corporation.

n players and s strategies each, the representational size is ns^n, 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 [14] such a compact representation, called *graphical game*, is defined as follows:

Definition 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 $N_G(p) \subseteq \{1, ..., n\}$ to denote the set of *neighbors of player* p in G – i.e., those vertices q such that the undirected edge (p, q) appears in G. By convention, $N_G(p)$ always includes p himself. 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 d of the graph, ns^d. If $|N_G(p)| = k$ and $s \in \prod_{i=1}^{k} S_i$, $M_p(s)$ denotes the payoff to p when his k neighbors (including himself) play s.

Consider a game with n players and strategy sets $S_1, ..., S_n$. 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 [10].

Definition 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.

[7] define a *best response* strategy as follows:

Definition 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, $s_p \in BR_{M_p}(s_{-p})$.

A visual example of a graphical game called *FRIENDS* and its PNE can be found in [10].

2.2 Quantum Annealing

Quantum annealing is a metaheuristic for solving complex optimization and decision problems [12]. 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,

$$\mathcal{H}(s) = \sum_i h_i x_i + \sum_{i<j} J_{ij} x_i x_j \tag{1}$$

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 ith qubit. The basic process of quantum annealing is to physically interpolate between an initial Hamiltonian H_I with 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 [19].

$$H(t) = s(t)H_I + (1 - s(t))H_P \tag{2}$$

If this transition is executed sufficiently slow, the probability to find the ground state of the problem Hamiltonian is close to 1 [1]. 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 [5] is mathematically equivalent and uses 0 and 1 for the spin variables [28]. The quantum annealer is as well designed to minimize the functional form of the QUBO:

$$\min x^t Q x \qquad \text{with } x \in \{0,1\}^n \tag{3}$$

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 [13]. In [18] the QUBO formulation for the Set Cover problem is given by:

$$H_A = A \sum_{\alpha=1}^{n} \left(1 - \sum_{m=1}^{N} x_{\alpha,m}\right)^2 + A \sum_{\alpha=1}^{n} \left(\sum_{m=1}^{N} m x_{\alpha,m} - \sum_{k:\alpha \in V_k} x_k\right)^2 \tag{4}$$

and

$$H_B = B \sum_{k=1}^{N} x_k \tag{5}$$

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 α is $m \geq 1$, and 0 otherwise. The first energy term imposes the constraints that for any given α exactly one $x_{\alpha,m}$ must be 1, since each element of U must be included a fixed number of times. The second term states, that the number of times that we claimed α was included is in fact equal to the number of subsets V_k we have included with α as an element. A is a penalty value, which is added on top of the solution energy, described by $H = H_A + H_B$, 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, as stated in Eq. 5. 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 [3,14,21,27]. However, some investigations in determining PNE in graphical and similar variations of games were made.

Daskalakis and Papadimitriou present a reduction from GG 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 [7].

In [11], 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 [23].

With quantum computing gaining more and more attention[2] 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.

[2] https://www.gartner.com/smarterwithgartner/the-cios-guide-to-quantum-computing/.

Since some NE algorithms are restricted to a certain graphical game structure, see for example [17], we want to emphasize, that our approach is able to work on every graphical game structure, even if the dependency graph is not connected.

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 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 [10]. 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 $\mathcal{B} = \{BR_{M_p} \mid (1 \leq p \leq n)\}$ and has the cardinality $C_{\mathcal{B}} = \sum_{p=1}^{n} |BR_{M_p}|$.

Example 1. In Table 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.

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

A	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

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 $\mathcal{B}$ of the classical phase, the following question arises:

"Is there a union of the combined best response strategies of $\mathcal{B}$ which results in a global strategy profile, under the premise that every player plays one of his best response actions?"—As stated in Definition 3 this would lead to a PNE.

This question resembles the Set Cover (SC) problem, stated in Sect. 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 $\mathcal{B}$). However, for our purposes we have to modify the given formulation in Eq. 4 as follows:

$$H = A \sum_{p=1}^{n} \left(1 - \sum_{j=1}^{|S_p|} \sum_{m=1}^{C_{\mathcal{B}}} x_{p,j,m} \right)^2 + A \sum_{p=1}^{n} \sum_{j=1}^{|S_p|} \left(\sum_{m=1}^{C_{\mathcal{B}}} m x_{p,j,m} - \sum_{k:p,j \in s_k} x_k \right)^2 + A \left(n - \sum_{k=1}^{C_{\mathcal{B}}} x_k \right)^2 \tag{6}$$

Equation 6 is quite similar to Eq. 4. However, an element α 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 Sect. 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 2, states that exactly n sets of $\mathcal{B}$ should be included to form the global strategy profile. This constraint implicitly ensures that every player is playing one of his best response strategies.

A 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 Sect. 2.2. This vector indicates which best response strategy of each player should be chosen to form a PNE.

Example 2. To demonstrate the function of the last energy term (constraint) of Eq. 6, an excerpt of *best response* strategy combinations of $\mathcal{B}$ (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.

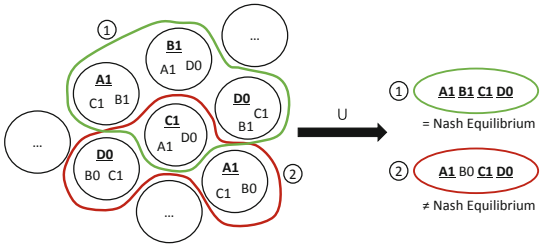

Fig. 1. On the left hand an excerpt of superset $\mathcal{B}$ is given. On the right hand the sets are united, with each (1) & (2) being a global strategy combination, but only one (1) being a PNE.

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 [11,15,23,29]. 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 [20] 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, [22]). 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*).

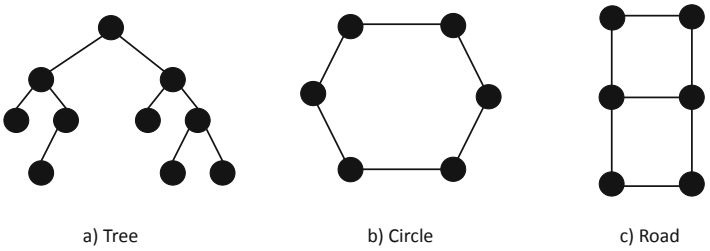

Fig. 2. Different graphical game structures (topologies) which indicate the dependencies between players of a game. (a) Tree-Topology with $n = 10$, (b) Circle-Topology with $n = 6$ and (c) Road-Topology with $n = 6$.

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 [4].

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, $\prod_{p=1}^{n} BR_{M_p}$.

Iterated Best Response: Additionally, an Iterated Best Response (IBR) algorithm was implemented [25]. In each iteration, one player changes his action to the best action that is the best response to the other players their action. This is repeated until a PNE is reached. In that case, the algorithm starts again from an randomly generated global strategy profile, to find other PNE. 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 Table 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 Fig. 3.

Determining best responses describes the time Q-Nash classical computing phase takes to identify the *best response* strategies of every player and additionally build up the QUBO matrix. The *Find embedding time* states the time D-Wave's classical embedding heuristic takes to find a valid subproblem hardware embedding. The *QBSolv time* can be divided into the *classical time* and the *quantum annealing time*. The *classical QBSolv time* 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* comprises of the number of subproblems times D-Wave's *qpu-access-time*.

[3] https://github.com/dwavesystems/qbsolv.

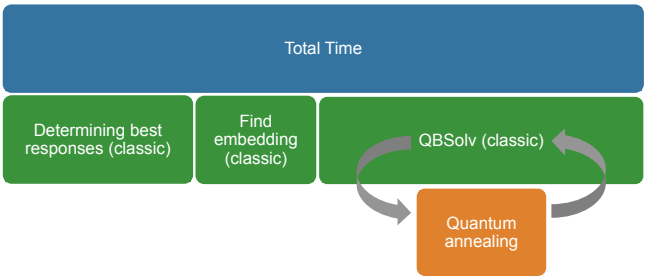

Fig. 3. Overview of the computational time components of Q-Nash

6 Results and Discussion

We investigated the solution quality and computational time of Q-Nash. Figure 4, 5 and 6 show the ability of finding PNE of the three proposed methods (Q-Nash with QBSolv, BF and IBR) in differently structured graphical games. For every graph topology (Tree, Circle and Road) we used games with players ranging from 6 to 30, due to the fact that the road topology needs an even number of players we skipped 15, 21 and 27 player instances. We ran Q-Nash and IBR 20 times on every instance. Additionally, IBR was run as long as Q-Nash (total time) took, to solve the instances.

The results show that Q-Nash with *num_repeats* set to 200, always found the same amount of PNE as the exact BF algorithm for the smaller game instances (6 to 12 Players, except of the 12 Player Tree-Topology instance), while IBR was only able to find all PNE in the Circle-Topology for those instances.

Regarding the larger game instances, one can see that Q-Nash, due being a heuristic, was not always able to find the same amount of PNE per run (20 times) and also was not able to find every PNE in a game, when compared to the exact BF algorithm. Compared to IBR, one can notice, that Q-Nash in general performed better than IBR on Circle-Topology, while IBR outperformed Q-Nash on tree structured games. Regarding the Road-Topology, Q-Nash did better on 12 and 18 Player instances, while IBR surpassed Q-Nash on 24 and 30 Player instances.

As already mentioned in Sect. 5.2 QBSolv splits the QUBO into smaller components (subQUBOS) of a predefined subproblem size, which are then solved independently of each other. This process is executed iteratively as long as there is an improvement and it can be defined using the QBSolv parameter *num_repeats*. This parameter determines the number of times to repeat the splitting of the QUBO problem matrix after finding a better sample. In Fig. 7 the influence of this parameter is shown. We exemplary used a 24 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 repeats (*num_repeats*) the inter-quartile range and its median in regard to the number of PNE found, increase. Although an annealing process takes only 20 µs in default,

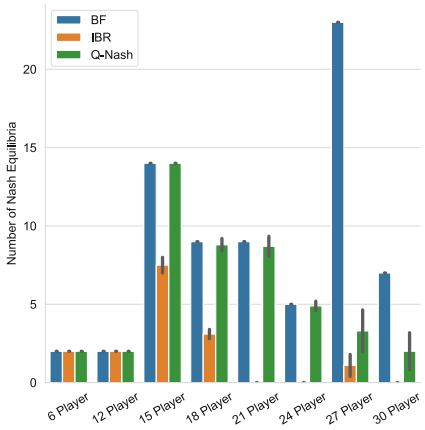

Fig. 4. Circle-Topology games

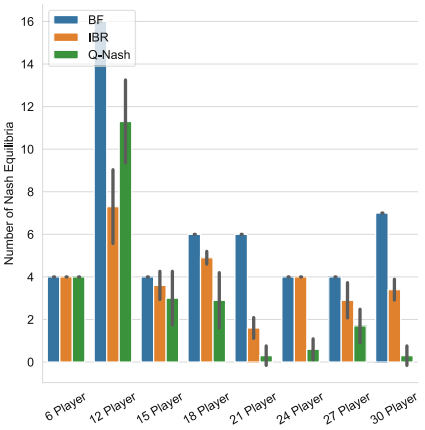

Fig. 5. Tree-Topology games

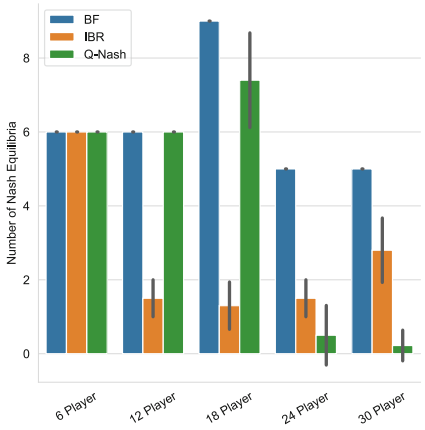

Fig. 6. Road-Topology games

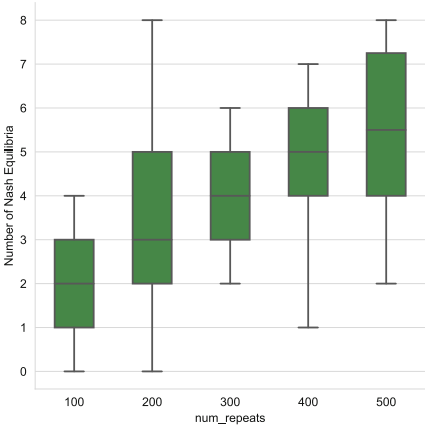

Fig. 7. The impact of *num_repeats* parameter on a 24 player road game.

it adds up with the *num_repeats* parameter and therefore leads to a trade-off between computational time and accuracy.

The computational time results are shown for circle structured graphical games and can be viewed in Table 2. The used game instances differ in the number of players (6–20 Player) and the computing times are given in seconds of our Q-Nash algorithm. As already mentioned in Sect. 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 Sect. 5.3.

Due to Q-Nash's immense overhead time (embedding, tabu search, latency, queuing time, etc.) the *total time (1)* is even for the smaller game instances (6 and 12 Players) quite large (85.393 and 125.828 s). Nevertheless the pure Q-Nash

Table 2. Computational times of the Q-Nash algorithm on various circle structured graphical games

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

computational time (*determining best response strategies (2) & QA time (5)*) is comparatively very small (1.359 and 1.754 s).

In general, one can see that the *quantum annealing time (5)* for these game instances is quite constant and the runtime of the *classical Q-Nash phase (4)* is only polynomial, as stated in Sect. 4.1.

With the *embedding time (3)* being constant, the only unpredictable runtime component of Q-Nash is *QBSolv classic (4)*. This is due to the tabu search (including the *num_repeats* parameter), 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 (2)*, the *embedding time (3)* and a fraction of the *quantum annealing time (5)*, since we do not have to split our problem instance into subproblems anymore. At this time, Q-Nash might be a competitive method regarding the computational time.

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 6–12 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 [2], a shorter total computation time could be achieved.

Regarding future work, it would also be interesting to see, how Q-Nash performs on a gate model quantum computer. For example, one could use the quantum approximate optimization algorithm, proposed by Farhi et al. [8]. This algorithm takes a QUBO Hamiltonian as an input. However since gate model quantum computers are at the moment even more limited in their resources, one has to come up with a clever way to split up large problem instances to fit them on the quantum chip.

References

1. Albash, T., Lidar, D.A.: Adiabatic quantum computation. Rev. Mod. Phys. **90**(1), 015002 (2018)
2. Aramon, M., Rosenberg, G., Valiante, E., Miyazawa, T., Tamura, H., Katzgraber, H.: Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. Bull. Am. Phys. Soc. **7**, 48 (2019)
3. Bhat, N.A., Leyton-Brown, K.: Computing Nash equilibria of action-graph games. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pp. 35–42. AUAI Press (2004)
4. Booth, M., Reinhardt, S.P., Roy, A.: Partitioning optimization problems for hybrid classical. quantum execution. Technical report, pp. 01–09 (2017)
5. Boros, E., Hammer, P.L., Tavares, G.: Local search heuristics for quadratic unconstrained binary optimization (QUBO). J. Heuristics **13**(2), 99–132 (2007)
6. Carfì, D., Musolino, F., et al.: Fair redistribution in financial markets: a game theory complete analysis. J. Adv. Stud. Financ. **2**(2), 4 (2011)
7. Daskalakis, C., Papadimitriou, C.H.: Computing pure Nash equilibria in graphical games via Markov random fields. In: Proceedings of the 7th ACM Conference on Electronic Commerce, pp. 91–99. ACM (2006)
8. Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014)
9. Fudenberg, D., Tirole, J.: Game Theory. MIT Press, Cambridge, vol. 392, no. 12, p. 80 (1991)
10. Gottlob, G., Greco, G., Scarcello, F.: Pure nash equilibria: hard and easy games. J. Artif. Intell. Res. **24**, 357–406 (2005)

[4] https://www.dwavesys.com/sites/default/files/mwj_dwave_qubits2018.pdf.

11. Jiang, A.X., Leyton-Brown, K.: Computing pure Nash equilibria in symmetric action graph games. In: AAAI, vol. 1, pp. 79–85 (2007)
12. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse Ising model. Phys. Rev. E **58**(5), 5355 (1998)
13. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of computer computations, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
14. Kearns, M., Littman, M.L., Singh, S.: Graphical models for game theory. arXiv preprint arXiv:1301.2281 (2013)
15. Koller, D., Milch, B.: Multi-agent influence diagrams for representing and solving games. Games Econ. Behav. **45**(1), 181–221 (2003)
16. La Mura, P.: Game networks. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 335–342 (2000)
17. Littman, M.L., Kearns, M.J., Singh, S.P.: An efficient, exact algorithm for solving tree-structured graphical games. In: Advances in Neural Information Processing Systems, pp. 817–823 (2002)
18. Lucas, A.: Ising formulations of many NP problems. Front. Phys. **2**, 5 (2014)
19. McGeoch, C.C.: Adiabatic quantum computation and quantum annealing: theory and practice. Synth. Lect. Quantum Comput. **5**(2), 1–93 (2014)
20. Nash, J.: Non-cooperative games. Ann. Math. **54**, 286–295 (1951)
21. Ortiz, L.E., Kearns, M.: Nash propagation for loopy graphical games. In: Advances in Neural Information Processing Systems, pp. 817–824 (2003)
22. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
23. Palmieri, A., Lallouet, A.: Constraint games revisited. In: International Joint Conference on Artificial Intelligence, IJCAI, vol. 2017, pp. 729–735 (2017)
24. Rabin, M.: Incorporating fairness into game theory and economics. Am. Econ. Rev. **83**, 1281–1302 (1993)
25. Roughgarden, T.: Twenty Lectures on Algorithmic Game Theory. Cambridge University Press, Cambridge (2016)
26. Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., Wu, Q.: A survey of game theory as applied to network security. In: 2010 43rd Hawaii International Conference on System Sciences (HICSS), pp. 1–10. IEEE (2010)
27. Soni, V., Singh, S., Wellman, M.P.: Constraint satisfaction algorithms for graphical games. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, p. 67. ACM (2007)
28. Su, J., Tu, T., He, L.: A quantum annealing approach for Boolean satisfiability problem. In: Proceedings of the 53rd Annual Design Automation Conference, p. 148. ACM (2016)
29. Vickrey, D., Koller, D.: Multi-agent algorithms for solving graphical games. In: AAAI/IAAI, pp. 345–351 (2002)
30. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)