Multi-agent Reinforcement Learning Using Simulated Quantum Annealing

Niels M. P. Neumann(✉), Paolo B. U. L. de Heer, Irina Chiscop, and Frank Phillipson

The Netherlands Organisation for Applied Scientific Research, Anna van Buerenplein 1, 2595 DA The Hague, The Netherlands
{niels.neumann,paolo.deheer,irina.chiscop,frank.phillipson}@tno.nl

Abstract. With quantum computers still under heavy development, already numerous quantum machine learning algorithms have been proposed for both gate-based quantum computers and quantum annealers. Recently, a quantum annealing version of a reinforcement learning algorithm for grid-traversal using one agent was published. We extend this work based on quantum Boltzmann machines, by allowing for any number of agents. We show that the use of quantum annealing can improve the learning compared to classical methods. We do this both by means of actual quantum hardware and by simulated quantum annealing.

Keywords: Multi-agent · Reinforcement learning · Quantum computing · D-Wave · Quantum annealing

1 Introduction

Currently, there are two different quantum computing paradigms. The first is gate-based quantum computing, which is closely related to classical digital computers. Making gate-based quantum computers is difficult, and state-of-the-art devices therefore typically have only a few qubits. The second paradigm is quantum annealing, based on the work of Kadowaki and Nishimore [17]. Problems have already been solved using quantum annealing, in some cases much faster than with classical equivalents [7,23]. Applications of quantum annealing are diverse and include traffic optimization [23], auto-encoders [18], cyber security problems [24], chemistry applications [12,28] and machine learning [7,8,21].

Especially the latter is of interest as the amount of data the world processes yearly is ever increasing [14], while the growth of the classical computing power is expected to stop at some point [27]. Quantum annealing might provide the necessary improvements to tackle these upcoming challenges.

One specific type of machine learning is reinforcement learning, where an optimal action policy is learnt through trial and error. Reinforcement learning can be used for a large variety of applications, ranging from autonomous robots [29] to determining optimal social or economical interactions [3]. Recently, reinforcement learning has seen many improvements, most notably the use of neural networks to encode the quality of state-action combinations. Since then, it has been successfully applied to complex games such as Go [25] and solving a Rubik's cube [2].

In this work we consider a specific reinforcement learning architecture called a Boltzmann machine [1]. Boltzmann machines are stochastic recurrent neural networks and provide a highly versatile basis to solve optimisation problems. However, the main reason against widespread use of Boltzmann machines is that the training times are exponential in the input size. In order to effectively use Boltzmann machines, efficient solutions for complex (sub)routines must be found. One of the complex subroutines is finding the optimal parameters of a Boltzmann machine. This task is especially well suited for simulated annealing, and hence for quantum annealing.

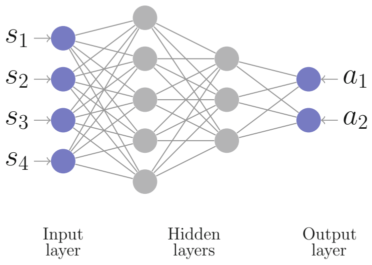
So far, little research has been done on quantum reinforcement learning. Early work demonstrated that applying quantum theory to reinforcement learning problems can improve the algorithms, with potential improvements to be quadratic in learning efficiency and exponential in performance [10,11]. Only recently, quantum reinforcement learning algorithms are implemented on quantum hardware, with [8] one of the first to do so. They demonstrated quantum-enabled reinforcement learning through quantum annealer experiments.

In this article, we consider the work of [8] and implement their proposed quantum annealing algorithm to find the best action policy in a gridworld environment. A gridworld environment, shown in Fig. 2, is a simulation model where an agent can move from cell to cell, and where potential rewards, penalties and barriers are defined for certain cells. Next, we extend the work to an arbitrary number of agents, each searching for the optimal path to certain goals. This work is, to our knowledge, the first simulated quantum annealing-based approach for multi-agent gridworlds. The algorithm can also be run on quantum annealing hardware if available.

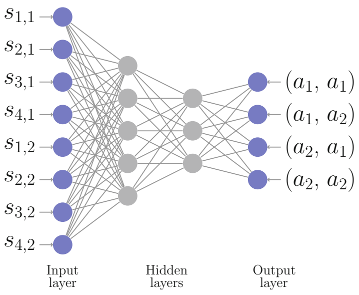
In the following section, we will give more details on reinforcement learning and Boltzmann machines. In Sect. 3 we will describe the used method and the extensions towards a multi-agent environment. Results will be presented and discussed in Sect. 4, while Sect. 5 gives a conclusion.

2 Background

A reinforcement learning problem is described as a *Markov Decision Process* (MDP) [6,15], which is a discrete time stochastic system. At every timestep t the agent is in a state s_t and chooses an action a_t from its available actions in that state. The system then moves to the next state s_{t+1} and the agent receives a reward or penalty $R_{a_t}(s_t, s_{t+1})$ for taking that specific action in that state. A policy π maps states to a probability distribution over actions and, when used as $\pi(s)$ it returns the highest-valued action a for state s. The policy will be optimized over the cumulative rewards attained by the agent for all state-action combinations. To find the optimal policy π^*, the Q-function $Q(s, a)$ is used which defines for each state-action pair the Q-value, denoting the expected cumulative reward, or the *quality*.



(a) A single-agent Boltzmann machine with four states and two actions.



(b) A multi-agent Boltzmann machine with two agents and per agent four states and two actions .

Fig. 1. Examples of restricted Boltzmann machines for reinforcement learning environments with one or more agents.

The Q-function is trained by trial and error, by repeatedly taking actions in the environment and updating the Q-values using the Bellman equation [5]:

$$Q^{\pi}(s_t, a_t) = \mathcal{E}_{s_{t+1}}\left[R_{a_t}(s_t, s_{t+1}) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))\right]. \tag{1}$$

Different structures can be used to represent the Q-function, ranging from a simple but very limited tabular Q-function to a (deep) neural network which encodes the values with the state vector as input nodes, and all possible actions as output nodes. In such deep neural networks, the link between nodes i and j is assigned a weight w_{ij}. These weights can then be updated using, for example, gradient descent, which minimizes a loss function. If a multi-layered neural network is used, it is called deep reinforcement learning (DRL). A special type of DRL is given by Boltzmann machines and their restricted variants.

A Boltzmann machine is a type of neural network that can be used to encode the Q-function. In a general Boltzmann machine, all nodes are connected to each other. In a restricted Boltzmann machine (RBM), nodes are divided into subsets of visible nodes v and hidden nodes h, where nodes in the same subset have no connections. The hidden nodes can be further separated in multiple hidden node subsets, resulting in a multi-layered (deep) RBM, an example of which can be seen in Fig. 1a with two hidden layers of 5 and 3 nodes respectively. There are also two visible layers. Connections between distinct nodes i and j are assigned a weight w_{ij}. Additionally, each node i is assigned a bias w_{ii}, indicating a preference to one of the two possible values ± 1 for that node. All links are bidirectional in RBMs, meaning $w_{ij} = w_{ji}$. Hence, they differ from feed-forward neural networks, where the weight of one direction is typically set to 0.

Using v_i for visible nodes and h_j for hidden ones, we can associate a global energy configuration to an RBM using

$$E(v, h) = -\sum_i w_{ii} v_i - \sum_j w_{jj} h_j - \sum_i \sum_j v_i w_{ij} h_j. \tag{2}$$

The probability for nodes being plus or minus one depends on this global energy and is given by

$$p_{\text{node } i=1} = \frac{1}{1+\exp(-\frac{\Delta E_i}{T})}.$$

Here $\Delta E_i = E_{\text{node } i=1} - E_{\text{node } i=-1}$ is the difference in global energy if node i is 1 or -1 and T is an internal model parameter, referred to as the temperature.

Simulated annealing can be used to update the weights w_{ij} quickly. In this approach, a subset of visible nodes are fixed (clamped) to their current values after which the network is sampled. During this process the anneal temperature is decreased slowly. This anneal parameter affects (decreases) the probability that the annealing process moves to a worse solution than the current one to avoid potential local minima. This sampling results in the convergence of the overall probability distribution of the RBM where the global energy of the network fluctuates around the global minimum.

3 Method

First, we will explain how the restricted quantum Boltzmann machine can be used to learn an optimal traversal-policy in a single-agent gridworld setting. Next, in Sect. 3.2 we will explain how to extend this model to work for a multi-agent environment.

3.1 Single-Agent Quantum Learning

In [8], an approach to a restricted quantum Boltzmann machine was introduced for a gridworld problem. In their approach, each state is assigned an input node and each action an output node. Additional nodes in the hidden layers are used to be able to learn the best state-action combinations. The topology for the hidden layers is a hyperparameter that is set before the execution of the algorithm. The task presented to the restricted Boltzmann machine is to find the optimal traversal-policy of the grid, given a position and a corresponding action.

Using a Hamiltonian associated to a restricted Boltzmann machine, we can find its energy. In its most general form, the Hamiltonian $\mathcal{H}_{\mathbf{v}}$ is given by

$$\mathcal{H}_{\mathbf{v}} = -\sum_{\substack{v\in V\\ h\in H}} w_{vh} v\sigma_h^z - \sum_{\{v,v'\}\subseteq V} w_{vv'} vv' - \sum_{\{h,h'\}\subseteq H} w_{hh'}\sigma_h^z\sigma_{h'}^z - \Gamma\sum_{h\in H}\sigma_h^x \tag{3}$$

with $\mathbf{v}$ denoting the prescribed fixed assignments of the visible nodes, i.e. the input and output nodes. Here V is the set of all visible nodes, while H is the set of all hidden nodes. Note that setting $w_{hh'} = 0$ has the same effect as removing the link between nodes h and h'. Also, Γ is an annealing parameter, while σ_i^z and σ_i^x are the spin-values of node i in the z- and x-direction, respectively. Note that in Eq. (3) no σ_v^z variables occur, as the visible nodes are fixed for a given sample, indicated by the v-terms. Note the correspondence between this Hamiltonian and the global energy configuration given in Eq. (2). The optimal traversal-policy is found by training the restricted Boltzmann machine, which means that the weights $w_{vv'}$, w_{vh} and $w_{hh'}$ are optimized based on presented training samples.

For each training step i, a random state s_i^1 is chosen which, together with a chosen action a_i^1, forms the tuple (s_i^1, a_i^1). Based on this state-action combination, a second state is determined: $s_i^2 \leftarrow a_i^1(s_i^1)$. The corresponding optimal second action a_i^2 can be found by minimizing the free energy of the restricted Boltzmann machine given by the Hamiltonian of Eq. (3). As no closed expression exists for this free energy, an approximate approach based on sampling $\mathcal{H}_{\mathbf{v}}$ is used.

For all possible actions a from state s_i^2, the Q-function corresponding to the RBM is evaluated. The action a that minimizes Q, is taken as a_i^2. Ideally, one would use the Hamiltonian $\mathcal{H}_{\mathbf{v}}$ from Eq. (3) for the Q-function. However, $\mathcal{H}_{\mathbf{v}}$ has both σ_h^x and σ_h^z terms that correspond to the spin of variable h in the x- and z-direction. As these two directions are perpendicular, measuring the state of one direction destroys the state of the other. Therefore, instead of $\mathcal{H}_{\mathbf{v}}$, we use an effective Hamiltonian $\mathcal{H}_{\mathbf{v}}^{eff}$ for the Q-function. In this effective Hamiltonian all σ_h^x terms are replaced by σ^z terms by using so-called *replica stacking* [20], based on the Suzuki-Trotter expansion of Eq. (3) [13,26].

With replica stacking, the Boltzmann machine is replicated r times in total. Connections between corresponding nodes in adjacent replicas are added. Thus, node i in replica k is connected to node i in replica $k \pm 1$ modulo r. Using the replicas, we obtain a new effective Hamiltonian $\mathcal{H}_{\mathbf{v}=(s,a)}^{eff}$ with all σ^x variables replaced by σ^z variables. We refer to the spin variables in the z-direction as $\sigma_{i,k}$ for node i in replica k and we identify $\sigma_{h,0} \equiv \sigma_{h,r}$. All σ^z variables can be measured simultaneously. Additionally, the weights in the effective Hamiltonian are scaled by the number of replicas. In its clamped version, i.e. with $v = (s, a)$ fixed, the effective resulting Hamiltonian $\mathcal{H}_{\mathbf{v}=(s,a)}^{eff}$ is given by

$$\begin{aligned}\mathcal{H}_{\mathbf{v}=(s,a)}^{eff} = &- \sum_{\substack{h\in H\\ h-s \text{ adjacent}}} \sum_{k=1}^{r} \frac{w_{sh}}{r}\sigma_{h,k} - \sum_{\substack{h\in H\\ h-a \text{ adjacent}}} \sum_{k=1}^{r} \frac{w_{ah}}{r}\sigma_{h,k} \\ &- \sum_{\{h,h'\}\subseteq H} \sum_{k=1}^{r} \frac{w_{hh'}}{r}\sigma_{h,k}\sigma_{h',k} - J^+ \sum_{h\in H}\sum_{k=0}^{r} \sigma_{h,k}\sigma_{h,k+1}. \qquad (4)\end{aligned}$$

Note that J^+ is an annealing parameter that can be set and relates to the original annealing parameter Γ. Throughout this paper, the values selected for Γ and J^+ are identical to those in [8].

For a single evaluation of the Hamiltonian and all corresponding spin variables, we get a specific spin configuration $\hat{h}$. We evaluate the circuit n_{runs} times for a fixed combination of s and a, which gives a multi-set $\hat{h}_{s,a} = \{\hat{h}_1, \ldots, \hat{h}_{n_{runs}}\}$ of evaluations. From $\hat{h}_{s,a}$, we construct a set of configurations $C_{\hat{h}_{s,a}}$ of unique spin combinations by removing duplicate solutions and retaining only one occurrence of each spin combination. Each spin configuration in $C_{\hat{h}_{s,a}}$ thus corresponds to one or more configurations in $\hat{h}_{s,a}$, and each configuration in $\hat{h}_{s,a}$ corresponds to precisely one configuration in $C_{\hat{h}_{s,a}}$.

The quality of $\hat{h}_{s,a}$, and implicitly of the weights of the RBM, is evaluated using the Q-function

$$Q(s,a) = -\left\langle \mathcal{H}^{eff}_{\mathbf{v}=(s,a)} \right\rangle - \frac{1}{\beta} \sum_{c \in C_{\hat{h}_{s,a}}} \mathbb{P}(c|s,a) \log \mathbb{P}(c|s,a), \tag{5}$$

where the Hamiltonian is averaged over all spin-configurations in $\hat{h}_{s,a}$. Furthermore, β is an annealing parameter and the frequency of occurrence of c in $\hat{h}_{s,a}$ is given by the probability $\mathbb{P}(c|s,a)$. The effective Hamiltonian $\mathcal{H}^{eff}_{\mathbf{v}=(s,a)}$ from Eq. (4) is used.

Using the Q-function from Eq. (5), the best action a_i^2 for state s_i^2 is given by

$$a_i^2 = \mathrm{argmin_a Q(s,\ a)} \tag{6}$$

$$= \mathrm{argmin_a} \left(-\left\langle \mathcal{H}^{\mathrm{eff}}_{\mathbf{v}=(s,a)} \right\rangle - \frac{1}{\beta} \sum_{\mathrm{c} \in \mathrm{C}_{\hat{\mathrm{h}}_{\mathrm{s,a}}}} \mathbb{P}(\mathrm{c|s,\ a}) \log \mathbb{P}(\mathrm{c|s,\ a}) \right). \tag{7}$$

Once the optimal action a_i^2 for state s_i^2 is found, the weights of the restricted Boltzmann machine are updated following

$$\Delta w_{hh'} = \epsilon \left(R_{a_i^1}\left(s_i^1, s_i^2\right) + \gamma Q\left(s_i^2, a_i^2\right) - Q\left(s_i^1, a_i^1\right) \right) \langle hh' \rangle, \tag{8}$$

$$\Delta w_{vh} = \epsilon \left(R_{a_i^1}\left(s_i^1, s_i^2\right) + \gamma Q\left(s_i^2, a_i^2\right) - Q\left(s_i^1, a_i^1\right) \right) v\langle h \rangle, \tag{9}$$

where, v is one of the clamped variables s_i^1 or a_i^1. The averages $\langle h \rangle$ and $\langle hh' \rangle$ are obtained by averaging the spin configurations in $\hat{h}_{s,a}$ for each h and all products hh' for adjacent h and h'. Based on the gridworld, a reward or penalty is given using the reward function $R_{a_i^1}(s_i^1, s_i^2)$. The learning rate is given by ϵ, and γ is a discount factor related to expected future rewards, representing a feature of the problem.

If the training phase is sufficiently long, the weights are updated such that the restricted Boltzmann machine gives the optimal policy for all state-action combinations. The required number of training samples depends on the topology of the RBM and the specific problem at hand. In the next section we will consider the extensions on this model to accommodate multi-agent learning.

3.2 Multi-agent Quantum Learning

In the previous section we considered a model with only a single agent having to learn an optimal policy in a grid, however, many applications involve multiple agents having conjoined tasks. For instance, one may think of a search-and-rescue setting where first an asset must be secured before a safe-point can be reached.

This model can be solved in different ways. First and foremost, different models can be trained for each task/agent involved. In essence, this is a form of multiple independent single-agent models. We will however focus on a model including *all* agents and *all* rewards simultaneously. This can be interpreted as one party giving orders to all agents on what to do next, given the states.

We consider the situation where the number of target locations is equal to the number of agents and each agent has to reach a target. The targets are not preassigned to the agents, however, each target can only be occupied by one agent simultaneously.

For this multi-agent setting, we extend the restricted Boltzmann machine as presented before. Firstly, each action of each agent is considered as an input state. For M agents, each with N actions, this gives MN input states. The output states are all possible combinations of the different actions of all agents. This means that if each agent has k possible actions, there are k^M different output states.

When using a one-hot encoding of states to nodes, the number of nodes in the network increases significantly compared to using a binary qubit encoding which allows for a more efficient encoding. Training the model using a binary encoding, however, is more complex than with one-hot encoding since for the former only a few nodes carry information on which states and actions are of interest while for binary encoding all nodes are used to encode the information. Therefore, we chose one-hot encoding similar to [8].

The Boltzmann machine for a multi-agent setting is closely related to that of a single-agent setting. An example is given in Fig. 1b for two agents. Here, input $s_{i,j}$ represents state i of agent j and output (a_m, a_n) means action m for the first agent and action n for the second.

Apart from a different RBM topology, also the effective Hamiltonian of Eq. (4) changes to accommodate the extra agents and the increase in possible action combinations for all agents. Again, all weights are initialized and state-action combinations denoted by tuples $(s_{i_1,1}, \ldots, s_{i_M,M}, (a_{i_1}, \ldots, a_{i_M}))$, are given as input to the Boltzmann machine. Let $a = (a_{i_1}, \ldots, a_{i_M})$ and $\mathcal{S} = \{s_{i_1,1}, \ldots, s_{i_M,M}\}$ and let r be the number of replicas. Nodes corresponding to these states and actions are clamped to 1, and other visible nodes are clamped to 0. The effective Hamiltonian is then given by

$$\begin{aligned}\mathcal{H}_{\mathbf{v}}^{eff}(v = (\mathcal{S}, a)) = & - \sum_{\substack{s \in \mathcal{S}, h \in H \\ h-s \text{ adjacent}}} \sum_{k=1}^{r} \frac{w_{sh}}{r} \sigma_{h,k} - \sum_{\substack{h \in H \\ h-a \text{ adjacent}}} \sum_{k=1}^{r} \frac{w_{ah}}{r} \sigma_{h,k} \\ & - \sum_{\{h,h'\} \subseteq H} \sum_{k=1}^{r} \frac{w_{hh'}}{r} \sigma_{h,k} \sigma_{h',k} - J^{+} \sum_{h \in H} \sum_{k=0}^{r} \sigma_{h,k} \sigma_{h,k+1}. \end{aligned} \tag{10}$$

In each training iteration a random state for each agent is chosen, together with the corresponding action. For each agent, a new state is determined based on these actions. The effective Hamiltonian is sampled n_{runs} times and the next best actions for the agents are found by minimizing the Q-function, with Eq. (10) used as effective Hamiltonian in Eq. (5). Next, the policy and weights of the Boltzmann machine are updated.

To update the weights of connections, Eq. (8) and Eq. (9) are used. Note that this requires the reward function to be evaluated for the entirety of the choices, consisting of the new states of all agents and the corresponding next actions.

4 Numerical Experiments

In this section the setup and results of our experiments are presented and discussed. First we explain how the models are sampled in Sect. 4.1, then the used gridworlds are introduced in Sect. 4.2. The corresponding results for the single-agent and multi-agent learning are presented and discussed in Sect. 4.3 and Sect. 4.4, respectively.

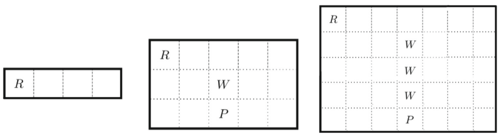
4.1 Simulated Quantum Annealing

There are various annealing dynamics [4] that can be used to sample spin values from the Boltzmann distribution resulting from the effective Hamiltonian of Eq. (3). The case $\Gamma = 0$ corresponds to purely classical simulated annealing [19]. Simulated annealing (SA) is also known as thermal annealing and finds its origin in metallurgy where the cooling of a material is controlled to improve its quality and correct defects.

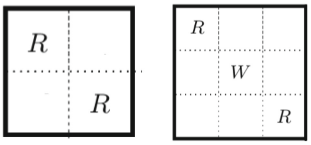
For $\Gamma \neq 0$, we have quantum annealing (QA) if the annealing process starts from the ground state of the transverse field and ends with a classical energy corresponding to the ground state energy of the Hamiltonian. The ground energy corresponds to the minimum value of the cost function that is optimized. No replicas are used for QA. The devices made by D-Wave Systems physically implement this process of quantum annealing.

However, we can also simulate quantum annealing using the effective Hamiltonian with replicas (Eq. (4)) instead of the Hamiltonian with the transverse field (Eq. (3)). This representation of the original Hamiltonian as an effective one, corresponds to simulated quantum annealing (SQA). Theoretically, SQA is a method to classically emulate the dynamics of quantum annealing by a quantum Monte Carlo method whose parameters are changed slowly during the simulation [22]. In other words, by employing the Suzuki-Trotter formula with replica stacking, one can simulate the quantum system described by the original Hamiltonian in Eq. (3).

Although SQA does not reproduce quantum annealing, it provides a way to understand phenomena such as tunneling in quantum annealers [16]. SQA can have an advantage over SA thanks to the capability to change the amplitudes of states in parallel, as proven in [9]. Therefore, we opted for SQA in our numerical experiments. We implemented the effective Hamiltonian on two different back-ends. The first using classical sampling given by simulated annealing (SQA SA). The second by implementing the effective Hamiltonian on the D-Wave 2000Q (SQA D-Wave 2000Q), a 2048 qubits quantum processor. Furthermore, we implemented a classical DRL algorithm for comparison.



(a) Gridworld problems considered in single-agent experiments.



(b) Gridworld problems considered in multi-agent experiments.

Fig. 2. All grids used to test the performance. On the left the three grids used for the single-agent scenario. On the right the grids used for multi-agent learning. The 3×3-grid is used with and without wall.

4.2 Gridworld Environments

We illustrate the free energy-based reinforcement learning algorithm proposed in Sect. 3.2 by applying it on the environments shown in Fig. 2a and Fig. 2b for one and two agents, respectively. In each gridworld, agents are allowed to take the actions up, down, left, right and stand still. We consider only examples with deterministic rewards. Furthermore, we consider environments both with and without forbidden states (walls) or penalty states. The goal of the agent is to reach the reward while avoiding penalty states. In case of multiple agents and rewards, each of the agents must reach a different reward. The considered multi-agent gridworlds focus on two agents. These environments are however easily extendable to an arbitrary number of agents.

The discount factor γ, explained in Sect. 3.1, was set to 0.8, similar to [8]. An agent reaching a target location is rewarded a value of 200, while ending up in a penalty state is penalized by -200. An extra penalty of -10 is given for each step an agent takes. As the rewards propagate through the network, the penalty assigned to taking steps is overcome. In the multi-agent case, a reward of 100 is given to each agent if each is at a different reward state simultaneously.

To assess the results of the multi-agent QBM-based reinforcement learning algorithm, we compare the learned policy for each environment with the optimal one using a fidelity measure. The optimal policy for this measure was determined logically thanks to the simple nature of these environments. As fidelity measure for the single-agent experiments, the formula from [20] is used. The fidelity at the i-th training sample for the multi-agent case with n agents is defined as

$$fidelity(i) = (T_r \times |S|^n)^{-1} \sum_{k=1}^{T_r} \sum_{s \in S^n} \mathbf{1}_{A(s,i,k) \in \pi^*(s)}. \tag{11}$$

Here, T_r denotes the number of independent runs for the method, $|S|$ denotes the total amount of states in the environment, π^* denotes the optimal policy and $A(s,i,k)$ denotes the action assigned at the k-th run and i-th training sample to the state pair s. Each state pair s is an n-tuple consisting of the state of each agent. This definition of fidelity for the multi-agent case essentially records the amount of state pairs in which all agents took the optimal actions over all runs.

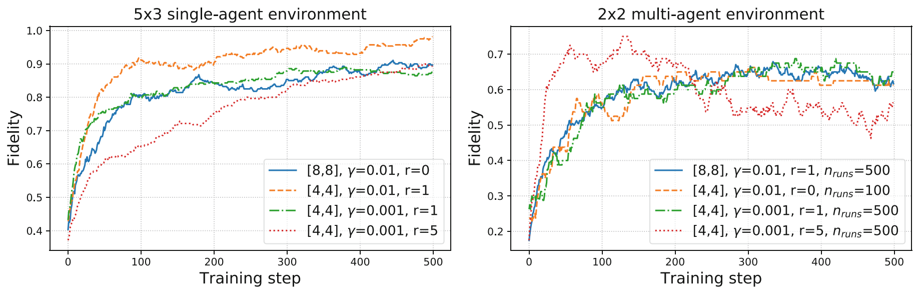


Fig. 3. Fidelity scores corresponding to hyperparameter choices with the highest average fidelity for the 5×3 single-agent and 2×2 multi-agent gridworld. The hyperparameters considered are the hidden layer size (described as an array of the number of nodes per layer), the learning rate γ and the number of replicas r. For the multi-agent grid search we also considered the number of samples per training step. The legends indicate the used values. (Color figure online)

4.3 Single-Agent Results

Before running the experiment, a grid search is performed to find the best setting for some hyperparameters. The parameters considered are: structure of the hidden layers, learning rate γ and number of replicas r used. These replicas are needed for the Suzuki-Trotter expansion of Eq. (3). The SQA SA reinforcement learning algorithm was run $T_r = 20$ times on the 5×3 grid shown in Fig. 2a for $T_s = 500$ training samples each run. In total, 18 different hyperparameter combinations are considered. For each, an average fidelity over all training steps is computed. The four best combinations are shown in the left plot of Fig. 3. Based on these results, the parameters corresponding to the orange curve (i.e. hidden layer size $= [4, 4], \gamma = 0.01, r = 1$) have been used in the experiments. These settings are used for all single-agent environments. The three different sampling approaches explained in Sect. 4.1 are used for each of the three environments. The results are all shown in Fig. 4.

We achieved similar results compared to the original single-agent reinforcement learning work in [8]. Our common means of comparison is the 5×3 gridworld problem, which in [8] also exhibits the best performance with SQA. Despite the fact that we did not make a distinction on the underlying graph of the SQA method, in our case the algorithm seems to achieve a higher fidelity within the first few training steps ($\sim$0.9 at the 100-th step in comparison to $\sim$0.6 in [8]) and to exhibit less variation in the fidelity later on in training. This may be due to the different method chosen for sampling the effective Hamiltonian.

Comparing sampling using SQA simulated annealing with SQA D-Wave 2000Q, we see the latter shows more variance in the results. This can be explained by the stochastic nature of the D-Wave system, the limited availability of QPU time in this research and the fact that only 100 D-Wave 2000Q samples are used at every training step. We expect that increasing the number of D-Wave 2000Q samples per training iteration increases the overall fidelity and results in a

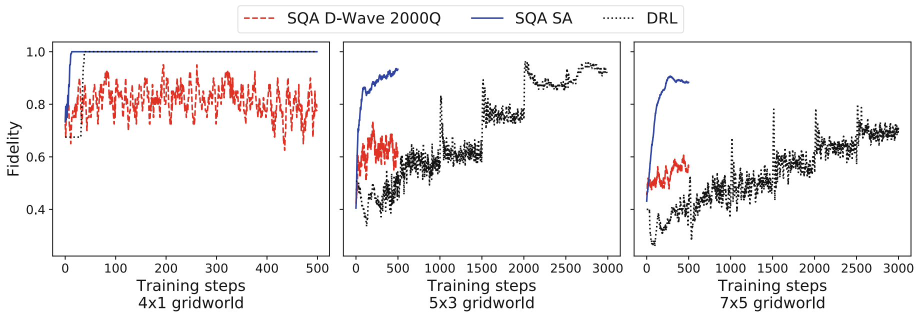


Fig. 4. The performance of the different RL implementations for the three single-agent gridworlds. All algorithms have been run $T_r = 10$ times.

smoother curve. However, it could also stem from the translation of the problem to the D-Wave 2000Q architecture. A problem that is too large to be directly embedded on the QPU is decomposed in smaller parts and the result might be suboptimal. This issue can be resolved by a richer QPU architecture or a more efficient decomposition.

Furthermore, the results could also be improved by a more environment specific hyperparameter selection. We now used the hyperparameters optimized for the 5×3 gridworld for each of the other environments. A gridsearch for each environment separately will probably improve the results. Increasing the number of training steps and averaging over more training runs will likely give a better performance and reduce variance for both SQA methods. Finally, adjusting the annealing schedule by optimizing the annealing parameter Γ could also lead to significantly better results.

Comparing the DRL to the SQA SA algorithm, we observe that the SQA SA algorithm achieves a higher fidelity using fewer training samples than the DRL for all three environments. Even SQA D-Wave 2000Q, with the limitations listed above, outperforms the classical reinforcement learning approach with exception of the 4×1 gridworld, the simplest environment. It is important to note that the DRL algorithm will ultimately reach a fidelity similar to both SQA approaches, but it does not reach this performance for the 5×3 and 7×5 gridworlds until having taken about six to twenty times as many training steps, respectively. Hence, the simulated quantum annealing approach on the D-Wave system learns more efficiently in terms of timesteps.

4.4 Multi-agent Results

As the multi-agent environments are fundamentally different from the single-agent ones, different hyperparameters might be needed. Therefore, we again run a grid search to find the optimal values for the same hyperparameters as in the single-agent case. Additionally, due to the complexity of the multi-agent environments, the number of annealing samples per training step $n_{runs} \in \{100, 500\}$ is also considered in the grid search.

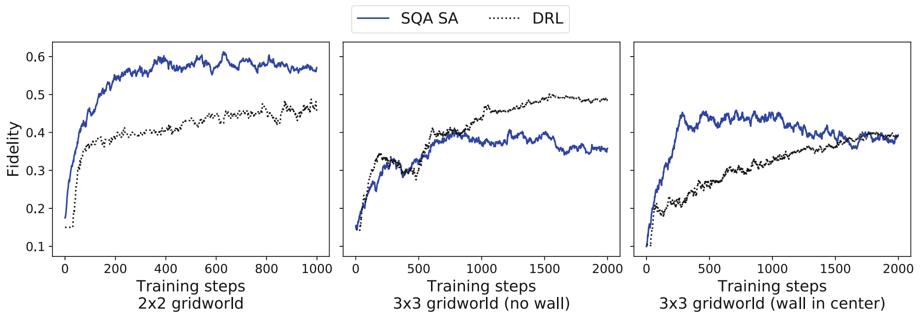


Fig. 5. Different RL methods for the three multi-agent gridworlds. All results are averaged over $T_r = 5$ runs.

For each combination of hyperparameters, the algorithm was run $T_r = 15$ times for $T_s = 500$ training samples each run for the 2×2 gridworld problem shown in Fig. 2b. In total, 36 combinations are considered and the performance of the best four combinations is given in the right plot in Fig. 3.

Based on the results from this gridsearch, suitable choices for the model parameters would either be given by the parameter sets corresponding to the green fidelity curve or the blue one. We opt for the blue fidelity curve, corresponding to a hidden layer topology of $[8, 8]$, a learning rate $\gamma = 0.01$, one replica and $n_{runs} = 500$ samples per training step, since that one is expected to be generalize better due to the larger hidden network and increased sampling.

The same hyperparameters found in the grid search conducted on the 2×2 gridworld problem are used for the two other environments. In Fig. 5, the results for the multi-agent environments are shown. As the available D-Wave 2000Q QPU time was limited in this research, only the results for the multi-agent SQA simulated annealing and the multi-agent DRL method are shown. An aspect that immediately stands out from the performance plots is the fast learning rate achieved by SQA SA within the first 250 training steps. In the case of classical DRL, learning progresses slower and the maximum fidelity reached is still lower than the best values achieved by SQA in the earlier iterations. We also see that the overall achieved fidelity is rather low for each of the environments compared to the single agent environments. This indicates that the learned policies are far from optimal. This can be due to the challenging nature of the small environments where multiple opposing strategies can be optimal, for instance, agent 1 moving to target 1 and agent 2 to target 2, and vice versa.

We expect the results for SQA D-Wave 2000Q to be better than the classical results, as SQA D-Wave 2000Q excels at sampling from a Boltzmann distribution, given sufficiently large hardware and sufficiently long decoherence times. We see that for two of the three environments, SQA SA learns faster and achieves at least a similar fidelity as classical methods. This faster learning and higher achieved fidelity is also expected of SQA D-Wave 2000Q.

5 Conclusion

In this paper we introduced free energy-based multi-agent reinforcement learning based on the Suzuki-Trotter decomposition and SQA sampling of the resulting effective Hamiltonian. The proposed method allows the modelling of arbitrarily-sized gridworld problems with an arbitrary number of agents. The results show that this approach outperforms classical deep reinforcement learning, as it finds policies with higher fidelity within a smaller amount of training steps. Some of the shown results are obtained using SQA simulated annealing, opposed to SQA quantum annealing which is expected to perform even better, given sufficient hardware and sufficiently many runs. Hence, a natural progression of this work would be to obtain corresponding results for SQA D-Wave 2000Q. The current architecture of the quantum annealing hardware is rather limited in size and a larger QPU is needed to allow fast and accurate reinforcement learning algorithm implementations of large problems.

Furthermore, implementing the original Hamiltonian without replicas on quantum hardware, thus employing proper quantum annealing, might prove beneficial. This takes away the need for the Suzuki-Trotter expansion and thereby a potential source of uncertainty. Moreover, from a practical point of view, it is worthwhile to investigate more complex multi-agent environments, where agents for instance have to compete or cooperate, or environments with stochasticity.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cogn. Sci. **9**(1), 147–169 (1985)
2. Agostinelli, F., McAleer, S., Shmakov, A., Baldi, P.: Solving the Rubik's cube with deep reinforcement learning and search. Nat. Mach. Intell. **1**, 356–363 (2019)
3. Arel, I., Liu, C., Urbanik, T., Kohls, A.: Reinforcement learning-based multi-agent system for network traffic signal control. IET Intell. Transp. Syst. **4**(2), 128–135 (2010)
4. Bapst, V., Semerjian, G.: Thermal, quantum and simulated quantum annealing: analytical comparisons for simple models. J. Phys. Conf. Ser. **473**, 012011 (2013)
5. Bellman, R.: On the theory of dynamic programming. Proc. Natl. Acad. Sci. **38**(8), 716–719 (1952)
6. Bellman, R.: A Markovian decision process. Indiana Univ. Math. J. **6**, 679–684 (1957)
7. Benedetti, M., Realpe-Gómez, J., Perdomo-Ortiz, A.: Quantum-assisted Helmholtz machines: a quantum-classical deep learning framework for industrial datasets in near-term devices. Quantum Sci. Technol. **3**(3), 034007 (2018)
8. Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J.S., Ronagh, P.: Reinforcement learning using quantum Boltzmann machines. CoRR 1612.05695
9. Crosson, E., Harrow, A.W.: Simulated quantum annealing can be exponentially faster than classical simulated annealing. In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS). IEEE, October 2016
10. Dong, D., Chen, C., Li, H., Tarn, T.J.: Quantum reinforcement learning. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **38**(5), 1207–1220 (2008)

11. Dunjko, V., Taylor, J.M., Briegel, H.J.: Quantum-enhanced machine learning. Phys. Rev. Lett. **117**(13), 130501 (2016)
12. Finnila, A., Gomez, M., Sebenik, C., Stenson, C., Doll, J.: Quantum annealing: a new method for minimizing multidimensional functions. Chem. Phys. Lett. **219**(5), 343–348 (1994)
13. Hatano, N., Suzuki, M.: Finding exponential product formulas of higher orders. In: Das, A., Chakrabarti, B.K. (eds.) Quantum Annealing and Other Optimization Methods. LNP, vol. 679, pp. 37–68. Springer, Heidelberg (2005). https://doi.org/10.1007/11526216_2
14. Hilbert, M., López, P.: The world's technological capacity to store, communicate, and compute information. Science **332**(6025), 60–65 (2011)
15. Howard, R.A.: Dynamic Programming and Markov Processes. Wiley for The Massachusetts Institute of Technology, Cambridge (1964)
16. Isakov, S.V., et al.: Understanding quantum tunneling through quantum Monte Carlo simulations. Phys. Rev. Lett. **117**(18), 180402 (2016)
17. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse Ising model. Phys. Rev. E **58**, 5355–5363 (1998)
18. Khoshaman, A., Vinci, W., Denis, B., Andriyash, E., Amin, M.H.: Quantum variational autoencoder. CoRR (2018)
19. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
20. Levit, A., Crawford, D., Ghadermarzy, N., Oberoi, J.S., Zahedinejad, E., Ronagh, P.: Free energy-based reinforcement learning using a quantum processor. CoRR (2017)
21. Li, R.Y., Felice, R.D., Rohs, R., Lidar, D.A.: Quantum annealing versus classical machine learning applied to a simplified computational biology problem. NPJ Quantum Inf. **4**(1), 1–10 (2018)
22. Mbeng, G.B., Privitera, L., Arceci, L., Santoro, G.E.: Dynamics of simulated quantum annealing in random Ising chains. Phys. Rev. B **99**(6), 064201 (2019)
23. Neukart, F., Compostella, G., Seidel, C., Dollen, D.V., Yarkoni, S., Parney, B.: Traffic flow optimization using a quantum annealer. Front. ICT **4**, 29 (2017)
24. Neukart, F., Dollen, D.V., Seidel, C.: Quantum-assisted cluster analysis on a quantum annealing device. Front. Phys. **6**, 55 (2018)
25. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
26. Suzuki, M.: Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. Commun. Math. Phys. **51**(2), 183–190 (1976). https://doi.org/10.1007/BF01609348
27. Waldrop, M.M.: The chips are down for Moore's law. Nature **530**(7589), 144–147 (2016)
28. Xia, R., Bian, T., Kais, S.: Electronic structure calculations and the Ising Hamiltonian. J. Phys. Chem. B **122**(13), 3384–3395 (2018)
29. Zhu, Y., et al.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3357–3364. IEEE (2017)