Reinforcement Learning Using Quantum Boltzmann Machines

Daniel Crawford[1], Anna Levit[1], Navid Ghadermarzy[1,2], Jaspreet S. Oberoi[1,3], and Pooya Ronagh[1,2,*]

Abstract. We investigate whether quantum annealers with select chip layouts can outperform classical computers in reinforcement learning tasks. We associate a transverse field Ising spin Hamiltonian with a layout of qubits similar to that of a deep Boltzmann machine (DBM) and use simulated quantum annealing (SQA) to numerically simulate quantum sampling from this system. We design a reinforcement learning algorithm in which the set of visible nodes representing the states and actions of an optimal policy are the first and last layers of the deep network. In absence of a transverse field, our simulations show that DBMs train more effectively than restricted Boltzmann machines (RBM) with the same number of weights. Since sampling from Boltzmann distributions of a DBM is not classically feasible, this is evidence of advantage of a non-Turing sampling oracle. We then develop a framework for training the network as a quantum Boltzmann machine (QBM) in the presence of a significant transverse field for reinforcement learning. This further improves the reinforcement learning method using DBMs.

Keywords. Reinforcement learning, Machine learning, Neuro-dynamic programming, Markov decision process, Quantum Monte Carlo simulation, Simulated quantum annealing, Restricted Boltzmann machine, Deep Boltzmann machine, General Boltzmann machine, Quantum Boltzmann machine, Quantum supremacy

1. Introduction

In view of recent advancements in the manufacturing of superconducting qubits [1, 2] and systems of qubits with second-order interactions, an important question for quantum computation is the existence of *quantum supremacy* (e.g., in [3]), that is, whether near-term quantum devices can outperform classical computers in *some*—in fact, *any*—computational task. With this motivation, we consider reinforcement learning as the computational task of interest, and design a method of reinforcement learning using a layout of quantum bits similar to that of a deep Boltzmann machine (DBM) (see Fig. 1b for a graphical representation). We use simulated quantum annealing (SQA) to demonstrate the advantage of reinforcement learning using quantum Boltzmann machines over its classical counterpart, in small problem instances.

Reinforcement learning ([4], known also as neuro-dynamic programming [5]) is an area of optimal control theory at the intersection of approximate dynamic programming and machine learning. It has been used successfully for many applications, in fields such as engineering [6, 7], sociology [8, 9], and economics [10, 11].

It is important to differentiate between reinforcement learning and common streams of research in machine learning. For instance, in supervised learning, the learning is facilitated by training samples provided by a source external to the agent and the computer. In reinforcement learning, the training samples are provided only by the interaction of the agent itself with the environment. For example, in a motion planning problem in an uncharted territory, it is desired that the agent learns in the fastest possible way to correctly navigate with the least number of blind decisions required to be taken. This is known as the dilemma of *exploration versus exploitation*; that is, neither exploration nor exploitation can be pursued exclusively without facing a penalty or failing at the task. The goal is hence not just to design an algorithm that eventually converges to an optimal policy, but for it to

[1] *1QB Information Technologies (1QBit), Vancouver, British Columbia, V6C 2B5, Canada*
[2] *Department of Mathematics, University of British Columbia, Vancouver, British Columbia, V6T 1Z2, Canada*
[3] *School of Engineering Science, Simon Fraser University, Burnaby, British Columbia, V5A 1S6, Canada*
[*] corresponding author: pooya.ronagh@1qbit.com

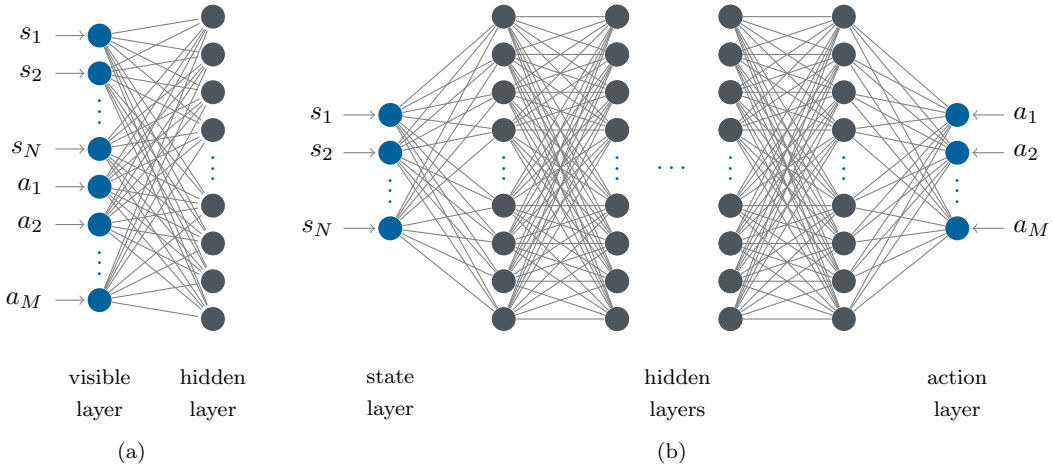


FIGURE 1. (a) The general RBM layout used in RBM-based reinforcement learning: the visible layer on the left consists of state and action nodes, and is connected to the hidden layer, forming a complete bipartite graph. (b) The general DBM layout used in DBM-based reinforcement learning: the visible nodes on the left represent states and the visible nodes on the right represent actions. The training procedure captures the correlations between states and actions in the weights of the edges between the nodes.

be able to generate good policies early in the learning process. We refer the reader to [4, Ch. 1.1] for a thorough introduction to the use cases and problem scenarios addressed by reinforcement learning.

The core idea in reinforcement learning is defining an operator on the Banach space of real-valued functions on the set of states of a system such that a fixed point of the operator carries information about an optimal policy of actions for a finite or infinite number of decision epochs. A numerical method for computing this fixed point is to explore this function space by travelling in a direction that minimizes the distance between two consecutive applications of the contraction mapping operator [5].

This optimization task, called *learning* in the context of reinforcement learning, can be performed by locally parametrizing the above function space using a set of auxiliary variables, and applying a gradient method to these variables. One approach for such a parametrization, due to [12], is to use the weights of a restricted Boltzmann machine (RBM) (see Fig. 1a) as the parameters, and the free energy of the RBM as an approximator for the elements in the function space. The descent direction is then calculated in terms of the expected values of the nodes of the RBM.

It follows from the universal approximation theorem [13] that RBMs can approximate any joint distribution over binary variables [14, 15]. However, in the context of reinforcement learning, RBMs are not necessarily the best choice for approximating *Q-functions* relating to Markov decision processes because [14] and [15] show that RBMs may require an exponential number of hidden variables with respect to the number of visible variables in order to approximate the desired joint distribution. On the other hand, DBMs have the potential to model higher-order dependencies than RBMs, and are more robust than deep belief networks [16].

One may, therefore, consider replacing the RBM with other graphical models and investigating the performance of the models in the learning process. Except in the case of RBMs, calculating statistical data from the nodes of a graphical model amounts to sampling from a Boltzmann distribution, creating a bottleneck in the learning procedure if performed classically [17].

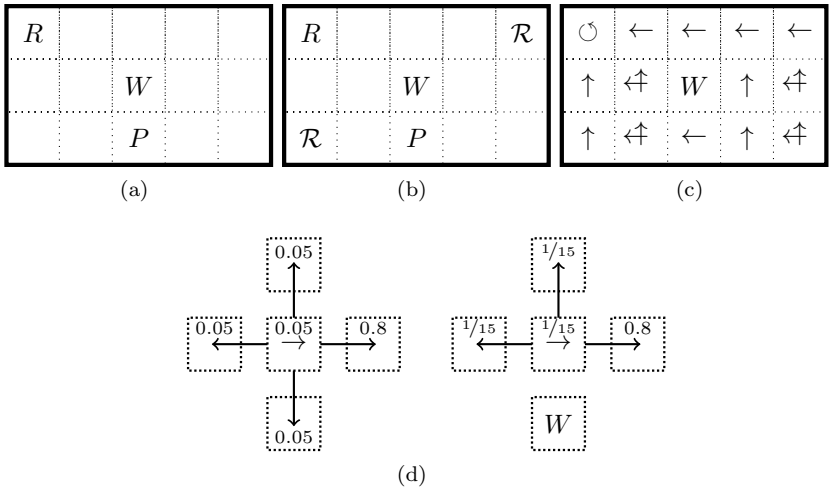


FIGURE 2. (a) A 3×5 maze. W represents a wall, R is a positive real number representing a reward, and P is a real number representing a penalty. (b) The previous maze with two additional stochastic rewards. (c) The set of all optimal actions for each cell of the maze in Fig (a). An optimal traversal policy is a choice of any combination of these actions. (d) A sample conditional state transition probability for a windy problem with no obstacles (left), and with a wall present (right)

As we explain in what follows, DBMs are good candidates for reinforcement learning tasks. Moreover, an important advantage of a DBM layout for a quantum annealing system is that the proximity and couplings of the qubits in the layout are similar to those of a sequence of bipartite blocks in D-Wave devices [18], and it is therefore feasible that such layouts could be manufactured in the near future. This is why, instead of attempting to embed a Boltzmann machine structure on an existing quantum annealing system as in [19, 20, 21, 22], we work under the assumption that the network itself is the native connectivity graph of a near-future quantum annealer, and, using numerical simulations, we attempt to understand its applicability to reinforcement learning.

Quantum Monte Carlo (QMC) numerical simulations have been found to be useful in simulating time-dependant quantum systems. Simulated quantum annealing (SQA) [23, 24], one of the many flavours of QMC methods, is based on the Suzuki–Trotter expansion of the path integral representation of the Hamiltonian of Ising spin models in the presence of a transverse field driver Hamiltonian. Even though the efficiency of SQA for finding the ground state of an Ising model is topologically obstructed [25], we consider the samples generated by SQA to be good approximations of the Boltzmann distribution of the quantum Hamiltonian [26]. Experimental studies have shown similarities in the behaviour of SQA and that of quantum annealing [27, 28] and its physical realization [29, 30] by D-Wave Systems.

We expect that when SQA is set such that the final strength of the transverse field is negligible, the distribution of the samples approaches the classical limit one expects to observe in absence of the transverse field. The classical counterpart of SQA is conventional simulated annealing (SA), which is based on thermal annealing. This algorithm can be used to create Boltzmann distributions from the Ising spin model only in the absence of a transverse field. It should, therefore, be possible to use SA or SQA to approximate the Boltzmann distribution of a classical Boltzmann machine. However, unlike SA, it is possible to use SQA not only to approximate the Boltzmann distribution of a classical Boltzmann machine, but also that of a graphical model in which the energy operator is a quantum Hamiltonian in the presence of a

transverse field. These graphical models, called quantum Boltzmann machines (QBM), were first introduced in [31].

We use SQA simulations to demonstrate evidence that a quantum annealing device that approximates the distribution of a DBM or a QBM may improve the learning process compared to a reinforcement learning method that uses classical RBM techniques. Other studies have shown that SQA is more efficient than thermal SA [23, 24]. Therefore, our method in conjunction with SQA can also be viewed as a quantum-inspired approach for reinforcement learning.

What distinguishes our work from current trends in quantum machine learning is that (i) we consider the use of quantum annealing in reinforcement learning applications rather than frequently studied classification or recognition problems; (ii) using SQA-based numerical simulations, we assume that the connectivity graph of a DBM directly maps to the native layout of a feasible quantum annealer; and (iii) the results of our experiments using SQA to simulate the sampling of an entangled system of spins suggest that using quantum annealers in reinforcement learning tasks has an advantage over thermal sampling.

2. Results

Maze traversal is a problem typically used to develop and benchmark reinforcement learning algorithms [32]. A maze is structured as a two-dimensional grid of r rows and c columns in which a decision-making agent is free to move up, down, left, right, or stand still. During this maze traversal, the agent encounters obstacles (e.g., walls), rewards (e.g., goals), and penalties (negative rewards, e.g., a pit). Each cell of the maze can contain either a deterministic or stochastic reward, a wall, a pit, or a neutral value. Fig. 2a and 2b show examples of two mazes. Fig. 2c shows the corresponding solutions to the maze in Fig. 2a.

The goal of the reinforcement learning algorithm in the maze traversal problem is for the agent to *learn* the optimal action to take in each cell of the maze by maximizing the total reward, that is, finding a route across the maze that avoids walls and pits while favouring rewards. This problem can be modelled as a Markov decision process (MDP) determined by the following components (see Sec. 5 for an overview on MDP formulations).

- The state of the system is the agent's position within the maze. The position state s takes values in the set of states

$$S = \{1, ..., r\} \times \{1, ..., c\}.$$

- In any state, the agent can decide to take one of the five actions

$$a \in \{\uparrow, \downarrow, \leftarrow, \rightarrow, \circlearrowleft\}.$$

  These actions will guide the agent through the maze. An action that leads the agent into a wall (W) or outside of the maze boundary is treated as an inadmissible action. Each action can be viewed as an endomorphism on the set of states

$$a : S \to S.$$

  If $a = \circlearrowleft$, then $a(s) = s$; otherwise, $a(s)$ is the state adjacent to S in the direction shown by a. We do not consider training samples where a is inadmissible.

- The transition kernel determines the probability of the agent moving from one state to another given a particular choice of action. In the simplest case, the probability of transition from s to $a(s)$ is one:

$$\mathbb{P}(a(s)|s, a) = 1.$$

  We call the maze *clear* if the associated transition kernel is as above, as opposed to the *windy* maze, in which there is a nonzero probability that if the action a is taken at state s, the next state will differ from $a(s)$.

- The immediate reward $r(s, a)$ that the agent gains from taking an action a in state s is the value contained in the destination state. Moving into a cell containing a reward returns the favourable value R, moving into a cell containing a penalty returns the unfavourable value P, and moving into a cell with no reward returns a *neutral* value in the interval (P, R).

- A discount factor for future rewards is a non-negative constant $\gamma < 1$. In our experiments, this discount factor is set to $\gamma = 0.8$. The discount factor is a feature of the problem rather than a free parameter of an implementation. For example, in a financial application scenario the discount factor might be a function of the risk-free interest rate.

The immediate reward for moving into a cell with a stochastic reward is given by a random variable $\mathcal{R}$. If an agent has prior knowledge of this distribution, then it should be able to treat the cell as one with a deterministic reward of value $\mathbb{E}[\mathcal{R}]$. This allows us to find the set of all optimal policies in each maze instance. This policy information is denoted by $\alpha^* : S \to 2^A$ associating with each state $s \in S$ a set of optimal actions $\alpha^*(s) \subseteq A$.

In our maze model, the neutral value is set to 100, the reward $R = 200$, and the penalty $P = 0$. In our experiments, the stochastic reward $\mathcal{R}$ is simulated by drawing a sample from the Bernoulli distribution $200\,\mathrm{Ber}(0.5)$; hence, it has the expected value $\mathbb{E}[\mathcal{R}] = 100$, which is identical to the neutral value. Therefore, the solutions depicted in Fig. 2c are solutions to the maze of Fig. 2b as well.

We study the performance of temporal-difference reinforcement learning algorithms (explained in detail in Sec. 5) using Boltzmann machines. We generalize the method introduced in [12], and compare the policies obtained from these algorithms to the optimal policy using a fidelity measure, which we define in (1).

The RBM reinforcement learning algorithm is due to Sallans and Hinton [12]. This algorithm uses the update rules (6) and (7) to update the weights of an RBM and (4) to calculate the expected values of random variables associated with the hidden nodes $\langle h \rangle$ (referred to as *activations* of the hidden nodes in machine learning terminology). As explained in Sec. 5.6, the main advantage of RBM is that it has explicit formulas for the hidden-node activations given the values of the visible nodes. Moreover, only for RBMs can the entropy portion of the free energy (26) be written in terms of the activations of the hidden nodes. More-complicated network architectures do not possess this property, so there is a need for a Boltzmann distribution sampler.

Since we are interested in the dependencies between states and actions, we consider a DBM architecture that has a layer of states connected to the first layer of hidden nodes, followed by multiple hidden layers, and a layer of actions connected to the final layer of hidden nodes (see Fig. 1). We demonstrate the advantages of this deep architecture trained using SQA and the derivation in Sec. 5.6 of the temporal-difference gradient method for reinforcement learning using general Boltzmann machines (GBM).

For T_r independent training runs of the same reinforcement learning algorithm, T_s training samples are used for reinforcement learning. The fidelity measure at the i-th training sample is defined by

$$\mathrm{fid}(i) = (T_r \times |S|)^{-1} \sum_{l=1}^{T_r} \sum_{s \in S} \mathbb{1}_{A(s,i,l) \in \alpha^*(s)}, \tag{1}$$

where $A(s, i, l)$ denotes the action assigned at the l-th run and i-th training sample to the state s. In our experiments, each algorithm is run 1440 times, and for each run of an algorithm, $T_s = 500$ training samples are generated.

Fig. 3a and 3b show the fidelity of the generated policies obtained from various reinforcement learning experiments on two clear 3×5 mazes. In Fig. 3a, the maze includes one reward, one wall, and one pit, and in Fig. 3b, the maze includes two stochastic rewards in addition. In these experiments, the training samples are generated by sweeping over the maze. Each sweep iterates over the maze elements in the same order. This explains the periodic behaviour of the fidelity curves (cf. Fig. 3c).

The curves labelled 'QBM-RL' represent the fidelity of reinforcement learning using QBMs. Sampling from the QBM is performed using SQA. All other experiments use classical Boltzmann machines as their graphical model. In the experiment labelled 'RBM-RL', the graphical model is an RBM, trained classically using formula (4). The remaining curve is labelled 'DBM-RL' for classical reinforcement learning using a DBM. In these experiments, sampling from configurations of the DBM is performed with SQA (with $\Gamma_f = 0.01$). The fidelity results of DBM-RL coincide closely with those of sampling configurations of the DBM using simulated annealing;

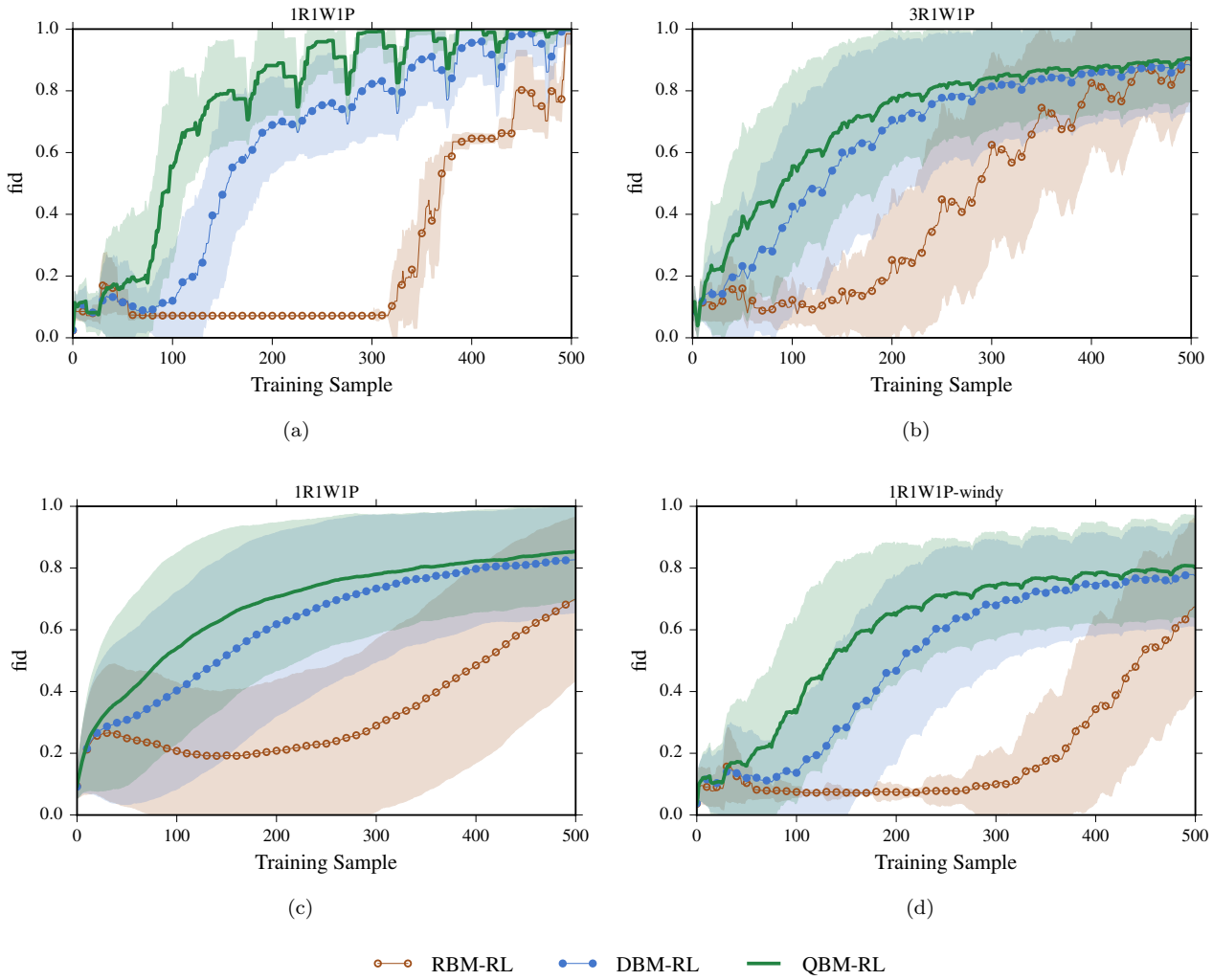
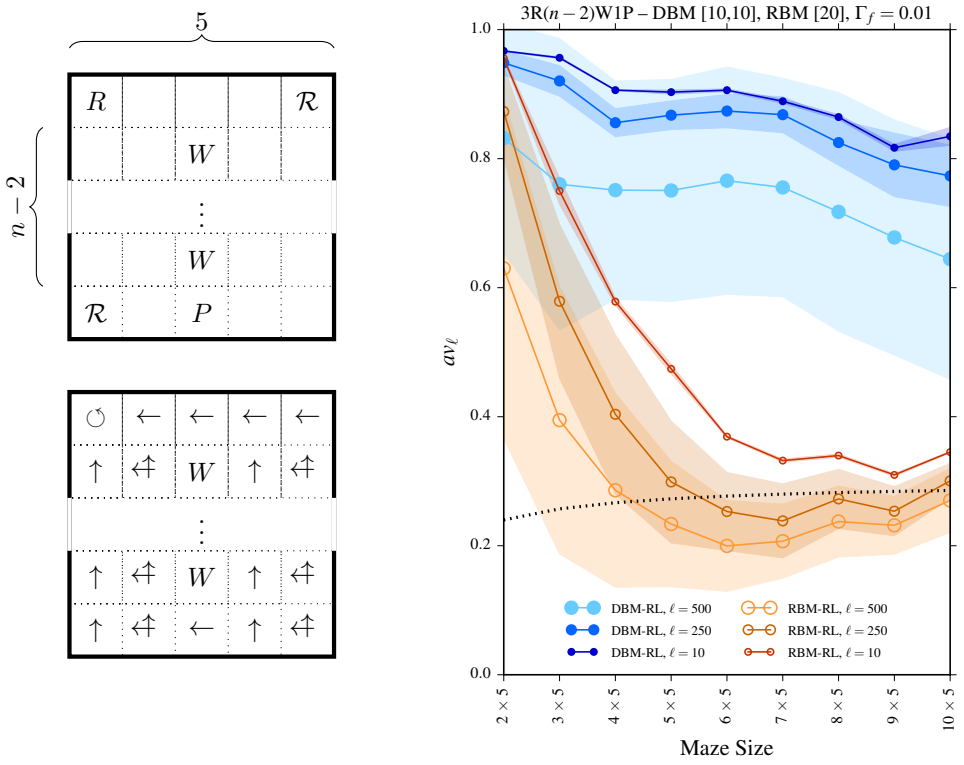


FIGURE 3. Comparison of RBM-RL, DBM-RL, and QBM-RL training results. Every underlying RBM has 16 hidden nodes and every DBM has two layers of 8 hidden nodes. The shaded areas indicate the standard deviation of each training algorithm. (a) The fidelity curves for the three algorithms run on the maze in Fig 2a. (b) The fidelity curves for the maze in Fig 2b. (c) The fidelity curves of the mentioned three algorithms corresponding to the same experiment as that of (a), except that the training is performed by uniformly generated training samples rather than sweeping the maze. (d) The fidelity curves corresponding to a windy maze similar to Fig 2a.

therefore, we have not sketched them. Fig. 3c regenerates the results of Fig. 3a using uniform random sampling (i.e., without sweeping through the maze).

Our next result, shown in Fig. 3d, compares RBM-RL, DBM-RL and QBM-RL for a windy maze of size 3×5. The transition kernel for this experiment is chosen such that $\mathbb{P}(a(s)|s,a) = 0.8$, and $\mathbb{P}(s'|s,a)$ has a nonzero value for all $s' \neq a(s)$ that are reachable from s by taking some action, in which case all of the values are equal. The transition probability is zero for all other states.



(a) (b)

FIGURE 4. A comparison between the performance of RBM-RL and DBM-RL as the size of the maze grows. All Boltzmann machines have 20 hidden nodes. (a) The schematics of an $n \times 5$ maze with one deterministic reward, one pit, two stochastic rewards, and $n-2$ walls. (b) The scaling of the average fidelity of each algorithm run on each instance of the $n \times 5$ maze. The dotted line is the average fidelity of uniformly randomly generated policies.

Fig. 2d shows examples of the transition probabilities in the windy problem.

To demonstrate the performance of RBM-RL and DBM-RL with respect to scaling, we define another measure called average fidelity, av_ℓ, where we take the average fidelity over the last ℓ training samples of the fidelity measure. Given T_s total training samples and $\mathrm{fid}(i)$ as defined above, we write

$$av_\ell = \frac{1}{\ell} \sum_{i=T_s-\ell}^{T_s} \mathrm{fid}(i)\,.$$

In Fig. 4, we report the effect of maze size on av_ℓ for RBM-RL and DBM-RL for varying maze sizes. We plot av_ℓ for RBM-RL and DBM-RL with $\ell = 500, 250$, and 10 as a function of maze size. We use nine $n \times 5$ mazes in this experiment indexed by various values of n. In addition to the av_ℓ plots, we include a dotted-line plot depicting the fidelity for a completely random policy.

3. Discussion

The fidelity curves in Fig. 3 show that the DBM-RL algorithm outperforms the RBM-RL algorithm with respect to the number of training samples. Therefore, we expect that in conjunction with a high-performance sampler of

Boltzmann distributions (e.g., a quantum or a quantum-inspired oracle taken as such), the DBM-RL algorithm improves the performance of reinforcement learning. The QBM-RL algorithm improves upon the DBM-RL results even further by taking advantage of sampling in presence of a significant transverse field.

In each experiment, the fidelity curves from DBM-RL produced using SQA with $\Gamma_f = 0.01$ match the ones produced using SA. This is consistent with our expectation that using SQA with $\Gamma \to 0$ produces samples from the same distribution as SA, namely, the Boltzmann distribution of the classical Ising Hamiltonian with no transverse field.

The best algorithm in our experiments is evidently QBM-RL using SQA with a significant final transverse field (in this case, $\Gamma_f = 2.00$). This is consistent with ideas found in [31] on sampling at *freeze-out* [33].

Fig. 3c shows that, whereas the maze can be solved with fewer training samples using ordered sweeps of the maze, the periodic behaviour of the fidelity curves is due to this periodic choice of training samples. This effect disappears once the training samples are chosen uniformly randomly.

Fig. 3d shows that the improvement in the learning of DBM-RL and QBM-RL algorithms persists in the case of more-complicated transition kernels. The same ordering of fidelity curves discussed earlier is observed: QBM-RL outperforms DBM-RL, and DBM-RL outperforms RBM-RL.

It is worth mentioning that, even though it may seem that more connectivity between the hidden nodes may allow a Boltzmann machine to capture more-complicated correlations between the visible nodes, the training process of the Boltzmann machine becomes more computationally involved. In our reinforcement learning application, an RBM with m hidden nodes, and $n = |S| + |A|$ visible nodes, has mn weights to train. A DBM with two hidden layers of equal size has $\frac{1}{4}m(2n + m)$ weights to train. Therefore, when $m < 2n$, the training of the DBM is in a domain of a lower dimension. Further, a GBM with all of its hidden nodes forming a complete graph requires $mn + \binom{m}{2}$ weights to train, which is always larger than that of an RBM or a DBM with the same number of hidden nodes.

One can observe from Fig. 4 that, as the maze size increases and the complexity of the reinforcement learning task increases, av_ℓ decreases for each algorithm. The RBM algorithm, while always outperformed by DBM-RL, shows a much faster decay in average fidelity as a function of maze size compared to DBM-RL. For larger mazes, the RBM algorithm fails to capture maze traversal knowledge, and approaches av_ℓ of a random action allocation (the dotted line), whereas the DBM-RL algorithm continues to be trained reasonably well. DBM-RL is capable of training the agent to traverse larger mazes, whereas the RBM algorithm, utilizing the same number of hidden nodes and a larger number of weights, fails to converge to an output better than a random policy. Given the ordering of fidelity curves discussed above, we expect QBM-RL to perform even better than DBM-RL as problem sizes increase, especially in the presence of a non-zero final transverse field.

The runtime and computational resources needed for DBM-RL and QBM-RL in comparison with RBM-RL are not investigated here. We expect that in view of [15] the size of the RBMs needed to solve larger maze problems will grow exponentially. It is, therefore, an interesting research path to pursue the extrapolation of the asymptotic complexity and size of the DBM-RL and QBM-RL algorithms in the quest for quantum supremacy.

4. Method

In this section, we present the details of classical reinforcement learning using RBM, a semi-classical approach base on a DBM (using SA and SQA), and quantum reinforcement learning (using SQA). Pseudo-code for these methods is provided in Algorithms 1, 2, and 3 below.

SQA methods are a class of quantum-inspired algorithms that perform discrete optimization by classically simulating the quantum tunnelling phenomena (see [34, p. 422] for an introduction). The algorithm used in this paper is a single spin-flip version of quantum Monte Carlo numerical simulation based on the Suzuki–Trotter formula, and uses the Metropolis acceptance probabilities. The SQA algorithm simulates the quantum annealing phenomena of an Ising spin model with a transverse field, that is,

$$\mathcal{H}(t) = -\sum J_{ij}\sigma_i^z\sigma_j^z - \sum h_i\sigma_i^z - \Gamma(t)\sum \sigma_i^x\,, \tag{2}$$

where σ^z and σ^x represent the Pauli z- and x-matrices, respectively, and time t ranges from 0 to 1. In this quantum evolution, the strength of the transverse field is slowly reduced to zero at finite temperature. In our implementations, we have used a linear transverse field schedule for the SQA algorithm as in [35] and [36]. Based on the Suzuki–Trotter formula, the key idea of this algorithm is to approximate the partition function of the Ising model with a transverse field as a partition function of a classical Hamiltonian denoted by $\mathcal{H}^{\text{eff}}$, corresponding to a classical Ising model of one dimension higher. More precisely,

(3)
$$\mathcal{H}^{\text{eff}}(\sigma) = -\sum_{\{i,j\}}\sum_{k=1}^{r}\frac{J_{ij}}{r}\sigma_{ik}\sigma_{jk} - J^{+}\sum_{i}\sum_{k=1}^{r}\sigma_{ik}\sigma_{i,k+1} - \sum_{i}\sum_{k=1}^{r}\frac{h_i}{r}\sigma_{ik}\,,$$

where r is the number of replicas, $J^{+} = \frac{1}{2\beta}\log\coth\left(\frac{\Gamma\beta}{r}\right)$, and σ_{ik} represent spins of the classical system of one dimension higher.

In our experiments, the strength Γ of the transverse field is scheduled to linearly decrease from 20.00 to one of $\Gamma_f = 0.01$ or 2.00. The inverse temperature β is set to the constant 2.00. Each spin is replicated 25 times to represent the Trotter slices in the extra dimension. The simulation is set to iterate over all replications of all spins one time per sweep, and the number of sweeps is set to 300. For each instance of input, the SQA algorithm is run 150 times. After termination, the configuration of each replica, as well as the configuration of the entire classical Ising model of one dimension higher, is returned.

Although the SQA algorithm does not follow the dynamics of a physical quantum annealer explicitly, it is used to simulate this process since it captures major quantum phenomena such as tunnelling and entanglement [27]. In [27], for example, it is shown that quantum Monte Carlo simulations can be used to understand the tunnelling behaviour in quantum annealers. As mentioned previously, it readily follows from the results of [26] that the limiting distribution of SQA is the Boltzmann distribution of $\mathcal{H}^{\text{eff}}$. This makes SQA a candidate classical algorithm for sampling from Boltzmann distributions of classical and quantum Hamiltonians. The former is achieved by setting $\Gamma_f \simeq 0$, and the latter by constructing an effective Hamiltonian of the system of one dimension higher, representing the quantum Hamiltonian with non-negligible Γ_f. Alternatively, a classical Monte Carlo simulation used to sample from the Boltzmann distribution of the classical Ising Hamiltonian is the SA algorithm, based on thermal fluctuations of classical spin systems.

Algorithm 1 RBM-RL

initialize weights of RBM
for all training samples (s_1, a_1) **do**
  $s_2 \leftarrow a_1(s_1)$, $a_2 \leftarrow \text{argmax}_a\, Q(s_2, a)$
  calculate $\langle h_i\rangle$ for $(i = 1, 2)$ using (4)
  calculate $F(\mathbf{s}_i, \mathbf{a}_i)$ for $(i = 1, 2)$ using (5)
  $Q(s_i, a_i) \leftarrow -F(\mathbf{s}_i, \mathbf{a}_i)$ for $(i = 1, 2)$
  update RBM weights using (6) and (7)
  $\pi(s_1) \leftarrow \text{argmax}_a\, Q(s_1, a)$
end for
return π

In Algorithm 1, we recall the steps of the classical reinforcement learning algorithm using an RBM with a graphical model similar to that shown in Fig. 1a. We set the initial Boltzmann machine weights using Gaussian zero-mean values with a standard deviation of 1.00, as is common practice for implementing Boltzmann machines [37]. Consequently, this initializes an approximation of a Q-function and a policy π given by

$$\pi(s) = \operatorname*{argmax}_a Q(s, a)\,.$$

We associate a classical spin variable σ_h to each hidden node h. Then the activations of the hidden nodes are calculated via

(4) $$\langle h\rangle = \mathbb{P}(\sigma_h = 1|\mathbf{s}, \mathbf{a}) = \sigma\left(\sum_{s\in S} w^{sh}s + \sum_{a\in A} w^{ah}a\right),$$

where $\sigma(\cdot)$ is the sigmoid function. Here $\mathbf{s}$ and $\mathbf{a}$ are encodings of state s and action a, respectively, using the state and action nodes of the Boltzmann machine. In our experiments, all Boltzmann machines have as many state nodes as $|S|$ and as many action nodes as $|A|$. We associate one node for every state $s \in S$, and the corresponding binary encoding is $\mathbf{s} = (0, 0, \ldots, 1, \ldots, 0)$, with

Algorithm 2 DBM-RL

1: initialize weights of DBM
2: **for all** training samples (s_1, a_1) **do**
3: $s_2 \leftarrow a_1(s_1)$, $a_2 \leftarrow \operatorname{argmax}_a Q(s_2, a)$
4: approximate $\langle h_i\rangle, \langle h_i h'_i\rangle, \mathbb{P}(\mathbf{h}|\mathbf{s}_i, \mathbf{a}_i)$ using SA or SQA for $(i = 1, 2)$
5: calculate $F(\mathbf{s}_i, \mathbf{a}_i)$ using (8) for $(i = 1, 2)$
6: $Q(s_i, a_i) \leftarrow -F(\mathbf{s}_i, \mathbf{a}_i)$ for $(i = 1, 2)$
7: update DBM weights using (6), (7), and (9)
8: $\pi(s_1) \leftarrow \operatorname{argmax}_a Q(s_1, a)$
9: **end for**
10: **return** π

zeroes everywhere except at the index of the node corresponding to s. We used similar encoding for the actions, using the action nodes. The free energy of the RBM is calculated using

$$(5)\quad -F(\mathbf{s},\mathbf{a}) = \sum_{\substack{s\in S\\ h\in H}} w^{sh} s\langle h\rangle + \sum_{\substack{a\in A\\ h\in H}} w^{ah} a\langle h\rangle - \sum_{h\in H} \left[\langle h\rangle \log\langle h\rangle + (1-\langle h\rangle)\log(1-\langle h\rangle)\right].$$

We refer the reader to Remark 5.1 or directly to [12] for more details. This results in an approximation of the Q-function (see Sec. 5.3) defined on the state–action space $S \times A$ [12]:

$$Q(s,a) \approx -F(\mathbf{s},\mathbf{a}).$$

We then use the update rules

$$(6)\quad \Delta w^{sh} = \varepsilon(r_n(s_n,a_n) + \gamma Q(s_{n+1},a_{n+1}) - Q(s_n,a_n)) s\langle h\rangle$$

and

$$(7)\quad \Delta w^{ah} = \varepsilon(r_n(s_n,a_n) + \gamma Q(s_{n+1},a_{n+1}) - Q(s_n,a_n)) a\langle h\rangle,$$

with a learning rate ε to update the weights of the RBM.

In Algorithm 2, we summarize the DBM-RL method. Here, the graphical model of the Boltzmann machine is similar to that shown in Fig. 1b. The initialization of the weights of the DBM is performed in a similar fashion to the previous algorithm. According to lines 4 and 5 of Algorithm 2, the samples from the SA or SQA algorithm are used to approximate the free energy of the classical DBM at points (s_1, a_1) and (s_2, a_2) using

$$(8)\quad -F(\mathbf{s},\mathbf{a}) = \sum_{\substack{s\in S\\ h\in H}} w^{sh} s\langle h\rangle + \sum_{\substack{a\in A\\ h\in H}} w^{ah} a\langle h\rangle + \sum_{\{h,h'\}\subseteq H} u^{hh'}\langle hh'\rangle - \frac{1}{\beta}\sum_{\mathbf{h}} \mathbb{P}(\mathbf{h}|\mathbf{s},\mathbf{a}) \log \mathbb{P}(\mathbf{h}|\mathbf{s},\mathbf{a}).$$

If SQA is used, averages are taken over each replica of each run; hence, there are 3750 samples of configurations of the hidden nodes for each state–action pair. The strength Γ of the transverse field is scheduled to linearly decrease from 20.00 to $\Gamma_f = 0.01$.

The SA algorithm is used with a linear inverse temperature schedule that increases from 0.01 to 2.00, in 50,000 sweeps, and is run 150 times. So, if SA is used, there are only 150 sample points used in the above approximation. The results of DBM-RL using SA or SQA match, with no significant difference.

The final difference between Algorithm 2 and Algorithm 1 is that the update rule now includes updates of weights between two hidden nodes given by

$$(9)\quad \Delta w^{hh'} = \varepsilon(r_n(s_n,a_n) + \gamma Q(s_{n+1},a_{n+1}) - Q(s_n,a_n))\langle hh'\rangle,$$

in addition to the previous rules 6 and 7 (see Sec. 5.6).

Algorithm 3 QBM-RL

1: initialize weights of DBM
2: **for all** training samples (s_1, a_1) **do**
3: $s_2 \leftarrow a_1(s_1)$, $a_2 \leftarrow \operatorname{argmax}_a Q(s_2, a)$
4: approximate $\langle h_i\rangle, \langle h_i h'_i\rangle, \langle \mathcal{H}^{\text{eff}}_{\mathbf{s_i},\mathbf{a_i}}\rangle$, and $\mathbb{P}(c|\mathbf{s_i}, \mathbf{a_i})$ using SQA for $(i = 1, 2)$
5: calculate $F(\mathbf{s}_i, \mathbf{a}_i)$ using (10) for $(i = 1, 2)$
6: $Q(s_i, a_i) \leftarrow -F(\mathbf{s}_i, \mathbf{a}_i)$ for $(i = 1, 2)$
7: update DBM weights using (6), (7), and (9)
8: $\pi(s_1) \leftarrow \operatorname{argmax}_a Q(s_1, a)$
9: **end for**
10: **return** π

The last algorithm is QBM-RL, presented in Algorithm 3. The initialization is performed as in Algorithms 1 and 2. However, according to lines 4 and 5, the samples from the SQA algorithm are used to approximate the free energy of a QBM at points (s_1, a_1) and (s_2, a_2) by computing the free energy corresponding to an effective classical Ising spin model of one dimension higher representing the quantum Ising spin model of the QBM,

$$F(\mathbf{s}, \mathbf{a}) = \langle \mathcal{H}^{\text{eff}}_{\mathbf{s},\mathbf{a}} \rangle + \frac{1}{\beta} \sum_c \mathbb{P}(c|\mathbf{s}, \mathbf{a}) \log \mathbb{P}(c|\mathbf{s}, \mathbf{a}) \,. \tag{10}$$

In this case, $\langle \mathcal{H}^{\text{eff}}_{\mathbf{s},\mathbf{a}} \rangle$ is approximated by the average energy of the entire system of one dimension higher and $\mathbb{P}(c|\mathbf{s}, \mathbf{a})$ is approximated by the normalized frequency of the configuration c of the entire system of one dimension higher (hence, there are only 150 sample points for each input instance in this case). The strength Γ of the transverse field in SQA is scheduled to linearly decrease from 20.00 to $\Gamma_f = 2.00$. In this algorithm, the weights are updated as in Algorithm 2. However, $\langle h \rangle$ and $\langle hh' \rangle$ in this algorithm represent expectations of measurements in the z-basis. See Sec. 5 for more details.

In each training iteration, we select a state–action pair $(s_1, a_1) \in S \times A$. Every node corresponding to a state or an action is removed from this graph and the configurations of the spins corresponding to the hidden nodes are sampled using SQA on an Ising spin model constructed as follows: the state s_1 contributes to a bias of $w^{s_1 h}$ to σ_h if h is adjacent to s_1; and the action a_1 contributes to a bias of $w^{a_1 h}$ to σ_h if h is adjacent to a_1. The bias on any spin σ_h for which h is a hidden node not adjacent to state s_1 or action a_1 is zero.

A subsequent state s_2 is obtained from a state–action pair (s_1, a_1) using the transition kernel outlined in Sec. 2, and a corresponding action a_2 is chosen via policy π. Another SQA sampling is performed in a similar fashion to the above for this pair.

In Fig. 3a, 3b, the selection of (s_1, a_1) is performed by sweeping over the set of state–action pairs. In Fig. 3d, the selection of (s_1, a_1) and s_2 is performed by sweeping over $S \times A \times S$. In Fig. 3c, the selection of s_1, a_1, and s_2 are all performed uniformly randomly.

We experiment with a variety of learning-rate schedules, including exponential, harmonic, and linear; however, we found that for the training of both RBMs and DBMs, an adaptive learning-rate schedule performed best (for information on adaptive subgradient methods, see [38]). In our experiments, the initial learning rate is set to 0.01.

In all of our studied algorithms, training terminates when a desired number of training samples have been processed, after which the updated policy is returned.

5. Supplementary Material

In this section, we derive the Q-learning method for Markov decision processes (MDP) with a function approximator represented by a general Boltzmann machine (GBM). To the best of our knowledge, this derivation has not been previously given, although it can be readily derived from the ideas presented in [12] and [31].

5.1. **Markov Decision Process.**

The stochastic control problem of interest to us is an MDP, defined as follows:

(i) finite sets of states S and of actions A;[1]

(ii) a controlled Markov chain [39], defined by a transition kernel $\mathbb{P}(s' \in S | s \in S, a \in A)$;[2]

(iii) a real-valued function $r : S \times A \to \mathbb{R}$ known as the *immediate reward structure*; and

(iv) a constant $\gamma \in [0, 1)$ known as the *discount factor*.

A function $\pi : S \to A$ is called a *stationary policy*; that is, it is a choice of action $\pi(s)$ for every state s independent of the point in time that the controlled process reaches s. A stationary policy π reduces the MDP into a time-homogeneous Markov chain, Π, with a transition probability $\mathbb{P}(s'|s, \pi(s))$.[3] The random process Π with initial condition $\Pi_0 = s$ we denote by Π^s.

Our *Markov decision problem* is to find

$$\pi^*(s) = \operatorname*{argmax}_{\pi \in A} V(\pi, s), \tag{11}$$

[1] When both S and A are finite, the MDP is said to be finite.

[2] The transition kernel does not need to be time-homogeneous; however, this definition suffices for the purposes of this work.

[3] For more-general statements, see [39].

where

$$V(\pi, s) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \, r\left(\Pi_i^s,\, \pi(\Pi_i^s)\right)\right]. \tag{12}$$

5.2. **Value Iteration.** Bellman [40] writes $V(\pi, s)$ recursively in the following manner using the monotone convergence theorem.

$$\begin{aligned} V(\pi, s) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \, r\left(\Pi_i^s,\, \pi(\Pi_i^s)\right)\right] \\ &= \mathbb{E}[r\left(\Pi_0^s,\, \pi(\Pi_0^s)\right)] + \gamma\, \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \, r\left(\Pi_{i+1}^s,\, \pi(\Pi_{i+1}^s)\right)\right] \\ &= \mathbb{E}[r\left(s,\, \pi(s)\right)] + \gamma \sum_{s' \in S} \mathbb{P}(s'|s, \pi(s))\, V(\pi, s') \end{aligned}$$

In particular,

$$\begin{aligned} V^*(s) &= V(\pi^*, s) \\ &= \max_a \left(\mathbb{E}[r\left(s,\, a\right)] + \gamma \sum_{s' \in S} \mathbb{P}(s'|s, a)\, V^*(s')\right). \end{aligned} \tag{13}$$

Hence, V^* is a fixed point for the operator

$$T(f) : s \mapsto \max_a \left(\mathbb{E}[r(s, a)] + \gamma \int f\right)$$

on the space $L_\infty(S)$ of bounded functions $S \to \mathbb{R}$ endowed with the max norm. Here, the integral is with respect to the probability measure on S, induced by the conditional probability distribution $\mathbb{P}(s'|s, a)$. It is easy to check that T is a contraction mapping and thus V^* is the unique fixed point of T and the uniform limit of any sequence of functions $\{T^k f\}_k$. Numerical computation of this limit using (13), called *value iteration*, is a common method of solving the Markov decision problem (11).

5.3. **Q-functions.** For a stationary policy π, the *Q-function* (also known as the *state–value function*) is defined by mapping a pair (s, a) to the expected value of the reward of the Markov chain that begins with taking action a at initial state s and continuing according to π [4]:

$$Q(\pi, s, a) = \mathbb{E}[r\left(s,\, a\right)] + \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \, r\left(\Pi_i^s,\, \pi(\Pi_i^s)\right)\right].$$

In particular, it is straightforward to check that

$$V(\pi, s) = \max_a Q(\pi, s, a),$$

and for $Q^*(s, a) = \max_\pi Q(\pi, s, a)$, the optimal policy for the MDP can be retrieved via the following:

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s, a). \tag{14}$$

This reduces the Markov decision problem to computing $Q^*(s, a)$. Through a Bellman-like recursion, we get

$$\begin{aligned} Q(\pi, s, a) &= \mathbb{E}[r\left(s, a\right)] + \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^i \, r\left(\Pi_i^s,\, \pi(\Pi_i^s)\right)\right] \\ &= \mathbb{E}[r\left(s, a\right)] + \gamma \sum_{s'} \mathbb{P}(s'|s, a) \max_{a'} Q(\pi, s', a'), \end{aligned}$$

which makes Q^* the fixed point of a different operator

$$T(f) : (s, a) \mapsto \mathbb{E}[r\left(s, a\right)] + \gamma \int \max_{a'} f$$

defined on $L_\infty(S \times A)$.

One can adapt a value iteration approach for Q similar to that for V. Even then, ε-optimal algorithms for this approach depend heavily on the cardinality of S and A, and suffer from the *curse of dimensionality* [40, 41].

5.4. **Temporal-Difference Gradient Descent.** From the previous section, we know that starting from an initial $Q_0 : S \times A \to \mathbb{R}$, the sequence $\{Q_n = T^n Q\}$ converges to Q^*. The difference

$$\begin{aligned} Q_{n+1}(s, a) - Q_n(s, a) &= \mathbb{E}[r(s, a)] \\ &\quad + \gamma \sum_{s'} \mathbb{P}(s'|s, a) \max_a \underbrace{Q_n(s', a)}_{(*)} - Q_n(s, a) \end{aligned} \tag{15}$$

is called the *temporal difference* of Q-functions, and is denoted by E_{TD}.

Employing a gradient approach to find the fixed point of T on $L_\infty(S \times A)$ involves locally parametrizing the functions in this space by a vector of parameters $\boldsymbol{\theta}$, that is,

$$Q(s,a) = Q(s,a;\boldsymbol{\theta}),$$

and travelling in the direction that minimizes $\|E_{\mathrm{TD}}\|^2$:

$$\Delta\boldsymbol{\theta} \propto -E_{\mathrm{TD}} \nabla_{\boldsymbol{\theta}} E_{\mathrm{TD}}\,. \tag{16}$$

The method TD(0) consists of treating the $(*)$ in (15) as constant with respect to the parameterization $\boldsymbol{\theta}$, in which case we may write

$$\Delta\boldsymbol{\theta} \propto\!\!\!\propto E_{\mathrm{TD}}(s,a) \nabla_{\boldsymbol{\theta}} Q(s,a;\boldsymbol{\theta}).$$

For an agent agnostic of the transition kernel or the distribution of the reward $r(s,a)$, this update rule for $\boldsymbol{\theta}$ is not possible. The alternative is to substitute, at each iteration, the expected value

$$\sum_{s'} \mathbb{P}(s'|s,a) \max_a Q_n(s_{n+1},a)$$

by $\max_a Q_n(s_{n+1},a)$, where s_{n+1} is drawn from the probability distribution $\mathbb{P}(s'|s,a)$, and substitute $\mathbb{E}[r(s_n,a_n)]$ by a sample of $r(s_n,a_n)$. This leads to a successful Monte Carlo training method called *Q-learning*.

In the following section, we explain the case where $\boldsymbol{\theta}$ comprises the weights of a Boltzmann machine.

5.5. **Clamped Boltzmann Machines.** A classical Boltzmann machine is a type of stochastic neural network with two sets V and H of visible and hidden nodes, respectively. Both visible and hidden nodes represent binary random variables. We use the same notation for a node and the binary random variable it represents. The interactions between the variables represented by their respective nodes are specified by real-valued weighted edges of the underlying undirected graph. A GBM, as opposed to models such as RBMs and DBMs, allows weights between any two nodes.

The *energy* of the classical Boltzmann machine is

$$\mathscr{E}(\mathbf{v},\mathbf{h}) = -\sum_{v\in V,\, h\in H} w^{vh} vh - \sum_{\{v,v'\}\subseteq V} w^{vv'} vv' - \sum_{\{h,h'\}\subseteq H} w^{hh'} hh', \tag{17}$$

with w^{vh}, $w^{vv'}$, and $w^{hh'}$ denoting the weights between visible and hidden, visible and visible, and hidden and hidden nodes of the Boltzmann machine, respectively, defined as a function of binary vectors $\mathbf{v}$ and $\mathbf{h}$ corresponding to the visible and hidden variables, respectively.

A clamped GBM is a neural network whose underlying graph is the subgraph obtained by removing the visible nodes for which the effect of a fixed assignment $\mathbf{v}$ of the visible binary variables contributes as constant coefficients to the associated energy

$$\mathscr{E}_{\mathbf{v}}(\mathbf{h}) = -\sum_{v\in V,\, h\in H} w^{vh} vh - \sum_{\{v,v'\}\subseteq V} w^{vv'} vv' - \sum_{\{h,h'\}\subseteq H} w^{hh'} hh'\,. \tag{18}$$

A clamped quantum Boltzmann machine (QBM) has the same underlying graph as a clamped GBM, but instead of a binary random variable, a qubit is associated to each node of the network. The energy function is substituted by the quantum Hamiltonian

$$\mathcal{H}_{\mathbf{v}} = -\sum_{v\in V,\, h\in H} w^{vh} v\sigma_h^z - \sum_{\{v,v'\}\subseteq V} w^{vv'} vv' - \sum_{\{h,h'\}\subseteq H} w^{hh'} \sigma_h^z \sigma_{h'}^z - \Gamma \sum_{h\in H} \sigma_h^x\,, \tag{19}$$

where σ_h^z represent the Pauli z-matrices and σ_h^x represent the Pauli x-matrices. Thus, a clamped QBM with $\Gamma = 0$ is equivalent to a clamped classical Boltzmann machine. This is because $\mathcal{H}_{\mathbf{v}}$ is a diagonal matrix in the σ^z-basis, the spectrum of which is identical to the range of $\mathscr{E}_{\mathbf{v}}$. The remainder of this section is formulated for the clamped QBMs, acknowledging that it can easily be specialized to clamped classical Boltzmann machines.

Let $\beta = \frac{1}{k_B T}$ be a fixed thermodynamic beta. For an assignment of visible variables $\mathbf{v}$, $F(\mathbf{v})$ denotes the *equilibrium free energy*, and is defined as

$$F(\mathbf{v}) := -\frac{1}{\beta} \ln Z_{\mathbf{v}} = \langle \mathcal{H}_{\mathbf{v}} \rangle + \frac{1}{\beta} \operatorname{tr}(\rho_{\mathbf{v}} \ln \rho_{\mathbf{v}}) \,. \tag{20}$$

Here, $Z_{\mathbf{v}} = \operatorname{tr}(e^{-\beta \mathcal{H}_{\mathbf{v}}})$ is the partition function of the clamped QBM and $\rho_{\mathbf{v}}$ is the density matrix $\rho_{\mathbf{v}} = \frac{1}{Z_{\mathbf{v}}} e^{-\beta \mathcal{H}_{\mathbf{v}}}$. The term $-\operatorname{tr}(\rho_{\mathbf{v}} \ln \rho_{\mathbf{v}})$ is the entropy of the system. The notation $\langle \cdots \rangle$ is used for the expected value of any observable with respect to the Gibbs measure, in particular,

$$\langle \mathcal{H}_{\mathbf{v}} \rangle = \frac{1}{Z_{\mathbf{v}}} \operatorname{tr}(\mathcal{H}_{\mathbf{v}} e^{-\beta \mathcal{H}_{\mathbf{v}}}).$$

5.6. **Reinforcement Learning with Clamped Boltzmann Machines.** Following the ideas in [12], the goal is to use the negative free energy of a Boltzmann machine to approximate the Q-function through the relationship

$$Q(s, a) \approx -F(\mathbf{s}, \mathbf{a}) = -F(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$$

for each admissible state–action pair $(s, a) \in S \times A$. Here, $\mathbf{s}$ and $\mathbf{a}$ are binary vectors encoding the state s and action a on the state nodes and action nodes, respectively, of the Boltzmann machine. Recall that in reinforcement learning, the visible nodes of the GBM are partitioned into two subsets of state nodes S and action nodes A.

The parameters $\boldsymbol{\theta}$, to be trained according to a TD(0) update rule (see Sec. 5.4), are the weights in a Boltzmann machine. For every weight w, the update rule is

$$\Delta w = -\varepsilon(r_n(s_n, a_n) + \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)) \frac{\partial F}{\partial w} \,.$$

From (20), we obtain

$$\begin{aligned} \frac{\partial F(\mathbf{s}, \mathbf{a})}{\partial w} &= \frac{1}{Z_{\mathbf{s},\mathbf{a}}} \frac{\partial}{\partial w} \operatorname{tr}(e^{-\beta \mathcal{H}_{\mathbf{s},\mathbf{a}}}) \\ &= -\frac{1}{Z_{\mathbf{s},\mathbf{a}}} \operatorname{tr}(\beta e^{-\beta \mathcal{H}_{\mathbf{s},\mathbf{a}}} \frac{\partial}{\partial w} \mathcal{H}_{\mathbf{s},\mathbf{a}}) \\ &= -\beta \left\langle \frac{\partial}{\partial w} \mathcal{H}_{\mathbf{s},\mathbf{a}} \right\rangle \,. \end{aligned}$$

Therefore, the update rule for TD(0) for the clamped QBM can be rewritten as

$$\begin{aligned} \Delta w^{vh} = \varepsilon(&r_n(s_n, a_n) \\ &+ \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)) v \langle \sigma_h^z \rangle \end{aligned} \tag{21}$$

and

$$\begin{aligned} \Delta w^{hh'} = \varepsilon(&r_n(s_n, a_n) \\ &+ \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)) \langle \sigma_h^z \sigma_{h'}^z \rangle. \end{aligned} \tag{22}$$

Here, h and h' denote two distinct hidden nodes and (by a slight abuse of notation) the letter v stands for a visible (state or action) node, and also the value of the variable associated to that node.

To approximate the right-hand side of each of (21) and (22), we use SQA experiments. By [42, Theorem 6], we may find the expected values of the observables $\langle \sigma_h^z \rangle$ and $\langle \sigma_h^z \sigma_{h'}^z \rangle$ by averaging the corresponding spins in the classical Ising model of one dimension higher constructed in SQA. To approximate the Q-function, we take advantage of [42, Theorem 4] and use (20) applied to this classical Ising model. More precisely, let $\mathcal{H}_{\mathbf{v}}^{\text{eff}}$ represent the Hamiltonian of the classical Ising model of one dimension higher and the associated energy function $\mathscr{E}_{\mathbf{v}}^{\text{eff}}$. The free energy of this model can be written

$$F(\mathbf{v}) = \langle \mathcal{H}_{\mathbf{v}}^{\text{eff}} \rangle + \frac{1}{\beta} \sum_c \mathbb{P}(c|\mathbf{v}) \log \mathbb{P}(c|\mathbf{v}) \,, \tag{23}$$

where c ranges over all spin configurations of the classical Ising model of one dimension higher.

The above argument holds in the absence of the transverse field, that is, for the classical Boltzmann machine. In this case, the TD(0) update rule is given by

$$\begin{aligned} \Delta w^{vh} = \varepsilon(&r_n(s_n, a_n) \\ &+ \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)) v \langle h \rangle \end{aligned} \tag{24}$$

and

$$\begin{aligned} \Delta w^{hh'} = \varepsilon(&r_n(s_n, a_n) \\ &+ \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)) \langle h h' \rangle \,, \end{aligned} \tag{25}$$

where $\langle h \rangle$ and $\langle h h' \rangle$ are the expected values of the variables and the product of variables, respectively,

in the binary encoding of the hidden nodes with respect to the Boltzmann distribution given by $\mathbb{P}(\mathbf{h}|\mathbf{v}) = \exp(-\mathscr{E}_{\mathbf{v}}(\mathbf{h}))/\sum_{\mathbf{h}'}\exp(-\mathscr{E}_{\mathbf{v}}(\mathbf{h}'))$. Therefore, they may be approximated using SA or SQA when $\Gamma \to 0$.

The values of the Q-functions in (24) and (25) can also be approximated empirically since in a classical Boltzmann machine,

(26)
$$\begin{aligned} F(\mathbf{v}) &= \sum_{\mathbf{h}} \mathbb{P}(\mathbf{h}|\mathbf{v})\mathscr{E}_{\mathbf{v}}(\mathbf{h}) + \frac{1}{\beta}\sum_{\mathbf{h}} \mathbb{P}(\mathbf{h}|\mathbf{v})\log\mathbb{P}(\mathbf{h}|\mathbf{v}) \\ &= -\sum_{\substack{s\in S\\ h\in H}} w^{sh}s\langle h\rangle - \sum_{\substack{a\in A\\ h\in H}} w^{ah}a\langle h\rangle - \sum_{\{h,h'\}\subseteq H} u^{hh'}\langle hh'\rangle \\ &\qquad + \frac{1}{\beta}\sum_{\mathbf{h}} \mathbb{P}(\mathbf{h}|\mathbf{s},\mathbf{a})\log\mathbb{P}(\mathbf{h}|\mathbf{s},\mathbf{a}). \end{aligned}$$

Remark 5.1. In the case of an RBM, Sallans and Hinton [12] show that the free energy is given by

$$\begin{aligned} -F(\mathbf{s},\mathbf{a}) &= \sum_{\substack{s\in S\\ h\in H}} w^{sh}s\langle h\rangle + \sum_{\substack{a\in A\\ h\in H}} w^{ah}a\langle h\rangle \\ &\quad - \sum_{h\in H}\left[\langle h\rangle\log\langle h\rangle + (1-\langle h\rangle)\log(1-\langle h\rangle)\right]. \end{aligned}$$

The update rule for the weights of the RBM is (24) alone. Moreover, in the case of RBMs, the equilibrium free energy $F(\mathbf{s},\mathbf{a})$ and its derivatives with respect to the weights can be calculated without the need for Boltzmann distribution sampling, according to the closed formula

$$\begin{aligned} \langle h\rangle = \mathbb{P}(\sigma_h = 1|\mathbf{s},\mathbf{a}) &= \sigma\left(\sum_{s\in S} w^{sh}s + \sum_{a\in A} w^{ah}a\right) \\ &= \left\{1+\exp\left(-\sum_{s\in S} w^{sh}s - \sum_{a\in A} w^{ah}a\right)\right\}^{-1}. \end{aligned} \tag{27}$$

Note that, in the general case, since the hidden nodes of a clamped Boltzmann machine are not independent, the calculation of the free energy is intractable.

Acknowledgements

We would like to thank Hamed Karimi, Helmut Katzgraber, Murray Thom, Matthias Troyer, and Ehsan Zahedinejad for reviewing this work and providing many helpful suggestions. The idea of using SQA to run experiments involving measurements with a non-zero transverse field was communicated in person by Mohammad Amin. We would also like to thank Marko Bucyk for editing this manuscript.

References

[1] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, pp. 194–198, 05 2011.

[2] J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I. C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and J. M. Martinis, "State preservation by repetitive error detection in a superconducting quantum circuit," *Nature*, vol. 519, pp. 66–69, 03 2015.

[3] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *arXiv:1608.00263v2*, 2016.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[5] D. Bertsekas and J. Tsitsiklis, *Neuro-dynamic Programming*. Anthropological Field Studies, Athena Scientific, 1996.

[6] V. Derhami, E. Khodadadian, M. Ghasemzadeh, and A. M. Z. Bidoki, "Applying reinforcement learning for web pages ranking algorithms," *Applied Soft Computing*, vol. 13, no. 4, pp. 1686–1692, 2013.

[7] S. Syafiie, F. Tadeo, and E. Martinez, "Model-free learning control of neutralization processes using reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 6, pp. 767–782, 2007.

[8] I. Erev and A. E. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *American Economic Review*, pp. 848–881, 1998.

[9] H. Shteingart and Y. Loewenstein, "Reinforcement learning and human behavior," *Current Opinion in Neurobiology*, vol. 25, pp. 93–98, 2014.

[10] T. Matsui, T. Goto, K. Izumi, and Y. Chen, "Compound reinforcement learning: theory and an application to finance," in *European Workshop on Reinforcement Learning*, pp. 321–332, Springer, 2011.

[11] Z. Sui, A. Gosavi, and L. Lin, "A reinforcement learning approach for inventory replenishment in vendor-managed inventory systems with consignment inventory," *Engineering Management Journal*, vol. 22, no. 4, pp. 44–53, 2010.

[12] B. Sallans and G. E. Hinton, "Reinforcement learning with factored states and actions," *JMLR*, vol. 5, pp. 1063–1088, Aug 2004.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[14] J. Martens, A. Chattopadhya, T. Pitassi, and R. Zemel, "On the representational efficiency of restricted Boltzmann machines," in *Advances in Neural Information Processing Systems*, pp. 2877–2885, 2013.

[15] N. Le Roux and Y. Bengio, "Representational power of restricted Boltzmann machines and deep belief networks," *Neural Computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[16] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines.," Apr 2009.

[17] P. M. Long and R. Servedio, "Restricted Boltzmann machines are hard to approximately evaluate or simulate," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 703–710, 2010.

[18] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S. Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose, "Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor," *Phys. Rev. B*, vol. 82, p. 024511, Jul 2010.

[19] M. Benedetti, J. Realpe-Gmez, R. Biswas, and A. Perdomo-Ortiz, "Quantum-assisted learning of graphical models with arbitrary pairwise connectivity," *arXiv:1609.02542*, 2016.

[20] S. H. Adachi and M. P. Henderson, "Application of Quantum Annealing to Training of Deep Neural Networks," *ArXiv e-prints*, Oct. 2015.

[21] M. Denil and N. de Freitas, "Toward the implementation of a quantum RBM," in *NIPS 2011 Deep Learning and Unsupervised Feature Learning Workshop*, 2011.

[22] M. Benedetti, J. Realpe-Gómez, R. Biswas, and A. Perdomo-Ortiz, "Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning," *Phys. Rev. A*, vol. 94, p. 022308, Aug. 2016.

[23] E. Crosson and A. W. Harrow, "Simulated Quantum Annealing Can Be Exponentially Faster than Classical Simulated Annealing," *ArXiv e-prints*, Jan. 2016.

[24] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, "Quantum versus classical annealing of Ising spin glasses," *Science*, vol. 348, pp. 215–217, Apr. 2015.

[25] M. B. Hastings and M. H. Freedman, "Obstructions to classically simulating the quantum adiabatic algorithm," *Quantum Information & Computation*, vol. 13, pp. 1038–1076, Nov 2013.

[26] S. Morita and H. Nishimori, "Convergence theorems for quantum annealing," *Journal of Physics A: Mathematical and General*, vol. 39, no. 45, p. 13903, 2006.

[27] S. V. Isakov, G. Mazzola, V. N. Smelyanskiy, Z. Jiang, S. Boixo, H. Neven, and M. Troyer, "Understanding quantum tunneling through quantum Monte Carlo simulations," *arXiv:1510.08057*, 2015.

[28] T. Albash, T. F. Rønnow, M. Troyer, and D. A. Lidar, "Reexamining classical and quantum models for the D-Wave One processor," *ArXiv e-prints*, Sept. 2014.

[29] L. T. Brady and W. van Dam, "Quantum Monte Carlo simulations of tunneling in quantum adiabatic optimization," *Phys. Rev. A*, vol. 93, no. 3, p. 032304, 2016.

[30] S. W. Shin, G. Smith, J. A. Smolin, and U. Vazirani, "How "quantum" is the D-Wave machine?," 2014.

[31] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum Boltzmann machine," *arXiv:1601.02036*, 2016.

[32] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," pp. 216–224.

[33] M. H. Amin, "Searching for quantum speedup in quasistatic quantum annealers," *Phys. Rev. A*, vol. 92, no. 5, 2015.

[34] S. M. Anthony Brabazon, Michael O'Neill, *Natural Computing Algorithms*. Springer-Verlag Berlin Heidelberg, 2015.

[35] R. Martoňák, G. E. Santoro, and E. Tosatti, "Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model," *Phys. Rev. B*, vol. 66, no. 9, p. 094203, 2002.

[36] B. Heim, T. F. Rønnow, S. V. Isakov, and M. Troyer, "Quantum versus classical annealing of Ising spin glasses," *Science*, vol. 348, no. 6231, pp. 215–217, 2015.

[37] G. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[38] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[39] S. Yuksel, "Control of stochastic systems," May 2016.

[40] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences*, vol. 42, no. 10, pp. 767–769, 1956.

[41] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[42] M. Suzuki, "Relationship between d-dimensional quantal spin systems and $(d+1)$-dimensional Ising systems equivalence, critical exponents and systematic approximants of the partition function and spin correlations," *Progress of Theoretical Physics*, vol. 56, no. 5, pp. 1454–1469, 1976.

[43] F. Abtahi and I. Fasel, "Deep belief nets as function approximators for reinforcement learning," *RBM*, vol. 2, p. h3, 2011.

[44] S. Elfwing, E. Uchibe, and K. Doya, "Scaled free-energy based reinforcement learning for robust and efficient learning in high-dimensional state spaces," *Value and Reward Based Learning in Neurobots*, p. 30, 2015.

[45] R. Martoňák, G. E. Santoro, and E. Tosatti, "Quantum annealing by the path-integral Monte Carlo method: The two-dimensional random Ising model," *Phys. Rev. B*, vol. 66, no. 9, p. 094203, 2002.

[46] M. Otsuka, J. Yoshimoto, and K. Doya, "Free-energy-based reinforcement learning in a partially observable environment.," *ESANN 2010 proceedings, European Symposium on Artificial Neural Networks – Computational Intelligence and Machine Learning*.

[47] J. Raymond, S. Yarkoni, and E. Andriyash, "Global warming: Temperature estimation in annealers," *arXiv:1606.00919*, 2016.

[48] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[49] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine learning*, vol. 38, no. 3, pp. 287–308, 2000.

[50] N. Frémaux, H. Sprekeler, and W. Gerstner, "Reinforcement learning using a continuous time actor-critic framework with spiking neurons," *PLoS Comput Biol*, vol. 9, no. 4, p. e1003024, 2013.

[51] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[52] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.