Hard-instance learning for quantum adiabatic prime factorization

Jian Lin,[1] Zhengfeng Zhang,[2] Junping Zhang,[2,*] and Xiaopeng Li[1,3,4,†]

[1]*State Key Laboratory of Surface Physics, Institute of Nanoelectronics and Quantum Computing, and Department of Physics, Fudan University, Shanghai 200433, China*

[2]*Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, Shanghai 200433, China*

[3]*Shanghai Qi Zhi Institute, Xuhui District, Shanghai 200032, China*

[4]*Shanghai Research Center for Quantum Sciences, Shanghai 201315, China*

(Received 7 April 2022; accepted 10 June 2022; published 29 June 2022)

Prime factorization is a difficult problem with classical computing, whose exponential hardness is the foundation of Rivest-Shamir-Adleman cryptography. With programable quantum devices, adiabatic quantum computing has been proposed as a plausible approach to solve prime factorization, having promising advantage over classical computing. Here, we find there are certain hard instances that are consistently intractable for both classical simulated annealing and unconfigured adiabatic quantum computing (AQC). Aiming at an automated architecture for optimal configuration of quantum adiabatic factorization, we apply a deep reinforcement learning (RL) method to configure the AQC algorithm. By setting the success probability of the worst-case problem instances as the reward to RL, we show the AQC performance on the hard instances is dramatically improved by RL configuration. The success probability also becomes more evenly distributed over different problem instances, meaning the configured AQC is more stable as compared to the unconfigured case. Through a technique of transfer learning, we find prominent evidence that the framework of AQC configuration is scalable—the configured AQC as trained on five qubits remains working efficiently on nine qubits with a minimal amount of additional training cost.

DOI: 10.1103/PhysRevA.105.062455

I. INTRODUCTION

Prime factorization plays a vital role in information security as its computation complexity on classical computers forms the foundation of Rivest-Shamir-Adleman (RSA) cryptography. The capability of factorizing N into a product of two prime integers $N = p\times q$ is enough to break the RSA cryptosystem. It has been attracting tremendous effort in a broad range of sciences from heuristic algorithm design [1] and machine learning [2,3] to bioinspired computation [4,5] and stochastic architectures [6]. Although this problem is not expected to be NP hard, all the established classical algorithms have an exponential time cost in the size of $\log N$, by which the RSA cryptosystem is secure. With quantum computing resources, Shor's quantum-circuit-based algorithm reduces the computation time cost to polynomial [7]. However, in the present era of noisy intermediate size quantum (NISQ) technology [8], the experimental implementation of Shor's quantum factorization is largely restricted to small integers [9,10] due to its demanding requirement on the qubit number and gate quality.

An alternative approach to perform prime factorization on quantum devices is through adiabatic quantum computing (AQC) [11,12], where the factorization problem is encoded into the ground state of a spin Hamiltonian $\mathbf{H}_\mathrm{P}$. To start, the quantum system is prepared in the ground state of a trivial Hamiltonian $\mathbf{H}_\mathrm{B}$ and then let evolve under a time (τ) -dependent Hamiltonian

$$\mathbf{H}(\tau) = [1 - \lambda(\tau/T)]\mathbf{H}_\mathrm{B} + \lambda(\tau/T)\mathbf{H}_\mathrm{P}, \tag{1}$$

where $\lambda(\tau/T)$ is the Hamiltonian schedule having $\lambda(0) = 0$ and $\lambda(1) = 1$ and T is the total quantum evolution time, or equivalently the AQC computation time.

The AQC model has been used to solve prime factorization by taking a cost function $(N - p\times q)^2$, with p and q in the binary representation and using a fixed Hamiltonian schedule as $\lambda(\tau/T) = (\tau/T)^2$ [12]. The classical binary-formed cost function is directly promoted to a quantum Hamiltonian $\mathbf{H}_\mathrm{P}$ using the quantum computation basis. One problem with this Hamiltonian encoding is the coupling strengths scale exponentially with $\log N$, which is unphysical. This problem is resolved with an improved encoding protocol incorporating the multiplication table [13,14]. This approach has received much attention in recent years [15–18], as triggered by the fascinating progress achieved in quantum annealing devices [19,20]. At the same time, concerns have been raised that the spin-glass problem arising in the spin Hamiltonian encoding may prohibit advantage in the AQC based prime factorization over classical solvers [19,21]. Here, we propose a scheme for AQC algorithm configuration based on reinforcement learning and apply it to prime factorization (see Fig. 1 for illustration). In the learning process, we use a reward setting that reflects the AQC performance on the most difficult factorization instances. Using a soft-actor-critic RL method, we find

*jpzhang@fudan.edu.cn
†xiaopeng_li@fudan.edu.cn

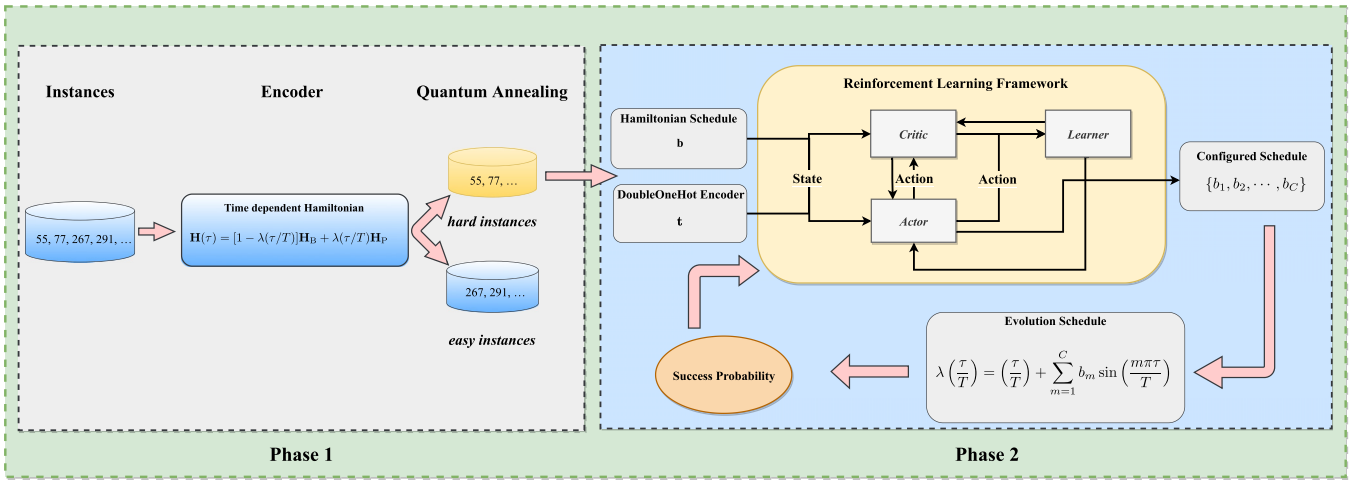

FIG. 1. Schematic illustration of hard instance learning architecture for quantum adiabatic prime factorization. In phase 1, we map the factorization problem to the quadratic unconstrained binary optimization (QUBO) problem and transfer it into an equivalent Ising type Hamiltonian (see Appendix A). We generate instances that need the same number of qubits to factorize. Then, we separate factorization instances into two groups according to their performance in AQC and load the intractable instances under the unconfigured AQC schedule into the RL optimization process. In phase 2, we show the RL structure. We combine the Hamiltonian schedule [all the b_m in Eq. (4)] and DoubleOneHot Encoder as the input state to the neural networks. The critic neural network supervises the actor neural network to take actions on the Hamiltonian schedule. A quantum adiabatic computer works under the configured schedule and provides the success probability as feedback to the RL framework.

the learning process has an astonishing convergence speed—it converges within only a few hundred measurement steps, significantly faster as compared to previous studies using RL for quantum state preparation [22], for parameter configuration in quantum approximate optimization [23], and for adiabatic quantum algorithm design [24], which takes about 10^4 to 10^6 measurement steps. The configured AQC algorithm produces an improved success probability, more evenly distributed over different factorization instances as compared with the unconfigured algorithm. Through the numerical test, we show the approach of RL-based AQC algorithm configuration has fair transferability.

II. EASY AND HARD INSTANCES IN THE FACTORIZATION PROBLEM

We map the prime factorization problem into the quadratic unconstrained binary optimization (QUBO) problem [25–27], which can be directly transferred into the Ising type Hamiltonian ($\mathbf{H}_P$) [14,18,28]. The details are provided in Appendix A. From the perspective of quantum spin glass, the major difficulty in reaching the ground state of a spin Hamiltonian is the existence of metastable spin configuration having large energy barriers [21,29]. Since the energy barriers cause slow relaxation in simulated annealing (SA) and make the algorithm inefficient, we group the different problem instances according to the efficiency of SA in reaching the true ground states.

Specifically, we take a classical spin system with energy defined by $\mathbf{H}_P/\|\mathbf{H}_P\|_\infty$, where $\|\mathbf{H}_P\|_\infty$ denotes the maximum coefficient in QUBO type cost function of all the instances in the same system size, and perform the following SA protocol. Starting from a random spin configuration corresponding to an infinite temperature ensemble, the spin configuration is locally updated at a randomly picked site, and then accepted with a probability $P(j) = e^{-\beta(j)\Delta E}$ at each step (labeled by j), where ΔE is the energy cost of the local spin update. The inverse temperature increases step by step according to $\beta(j) = \beta_0 e^{(j/j_0)}$, $j \in [0, 10j_0]$. With a given j_0, there is a corresponding success probability to reach the actual ground state at the end of SA. The success probability tends to increase with j_0, since a slower SA has a better chance to succeed. For each problem instance, we perform 500 parallel runs of SA. In each run, we increase j_0 until the actual ground state is reached. The corresponding j_0 is defined to be j_0^*. This quantity measures the number of steps for SA to succeed and can thus be used to quantify the energy barrier of the encoding spin-glass problem and consequently the hardness of the computation. We obtain the statistics of j_0^* out of the parallel SA runs. In Fig. 2(a), we show the mean values of j_0^* obtained in performing SA on factorizing 55, 65, 77, 91, 267, 291, 303, 309, 321, 327, 339, and 381, which involves 7 bits in our Hamiltonian encoding. The numbers 77 and 91 are found to be difficult to factorize, as the associated j_0^* is typically larger than other numbers.

We also test the different performances of problem instances with AQC. The time evolution of the quantum system is formally described by a time-dependent quantum state

$$|\psi(\tau)\rangle = \mathcal{T}_\tau e^{-i\int_0^\tau d\tau' \mathbf{H}(\tau')}|\psi(0)\rangle, \tag{2}$$

where $\mathcal{T}_\tau$ represents time ordering. The computation results of AQC are collected by collapsing the final quantum state in the computation basis. The results are stochastic in general. The success probability of AQC to collapse onto the correct solution tends to increase as we increase the adiabatic evolution time (T) [17]. In the numerical test, we increase T until the average success probability reaches a threshold P_{th}, which is set at 0.1. We separate the factorization instances into easy

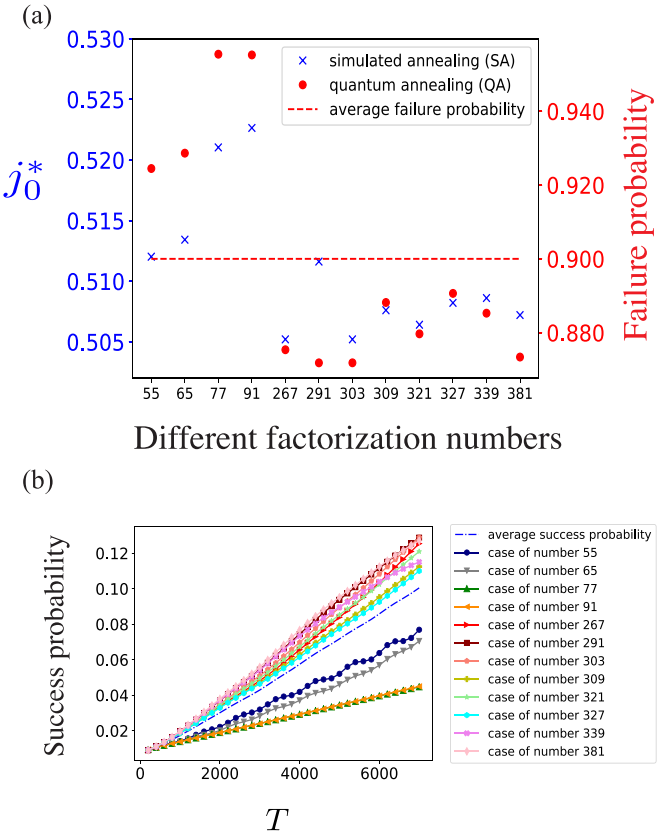

FIG. 2. Easy and hard factorization instances with simulated annealing and adiabatic quantum computing. We separate the problem instances into two groups according to their performances. In (a), blue data show the j_0^* obtained in 500 parallel runs of simulated annealing (see main text). Red data show the failure probability of unconfigured AQC (see main text) with a total evolution time $T = 6964$. The red dashed line denotes the average failure probability of all the problem instances. It reaches the threshold of failure probability which we set to be 0.9. The problem instances that are hard with SA which have a high value of j_0^* remain hard for the unconfigured AQC as it yields a considerably higher failure probability compared to other problem instances. In (b), we show the performances of different instances with the increasing total evolution time T. We find the success probabilities of several instances increase more slowly than others and are under the average value at the end. We separate them into two groups with the threshold (see main text). In these plots, we choose numbers to factorize that are encoded by seven spins.

and hard groups according to their individual AQC success probabilities. The problem instances having a success probability larger (smaller) than the threshold (P_{th}) are counted as easy (hard).

From Eq. (2), the final quantum state remains intact as we rescale $\tau \to \zeta\tau$, and $\mathbf{H} \to \mathbf{H}/\zeta$, with ζ an arbitrary scaling factor. For physical consideration, a Hamiltonian convention is imposed by

$$\mathbf{H}(\tau) \to \mathbf{H}(\tau)/\|\mathbf{H}_{\text{P}}\|_{\infty}, \tag{3}$$

where $\|\mathbf{H}_{\text{P}}\|_{\infty}$ denotes the same meaning in simulated annealing. With this convention, the required interaction strength in a physical system is guaranteed to be bounded.

We have checked the performance of the unconfigured AQC with a previously used Hamiltonian schedule $\lambda(\tau/T) = (\tau/T)^2$ [12]. As shown in Fig. 2(a), there are several factorization instances whose failure probabilities are substantially larger than other typical instances, e.g., with $N = 77, 91$. As shown in Fig. 2(b), the success probabilities of several cases increase quite slowly with T. The factorization problem instances that are hard with SA appear to remain hard on the unconfigured AQC.

III. ADIABATIC QUANTUM ALGORITHM CONFIGURATION BY REINFORCEMENT LEARNING

Aiming at rescuing the hard factorization instances, we develop an RL-based configuration scheme for the AQC algorithm (see Fig. 1 for illustration). It is known in the quantum adiabatic Grover search that different choices of Hamiltonian schedule lead to different computation complexity [24,30]. There are different choices for configuring the AQC algorithm in principle [31–34]. Here we choose to vary the standard Hamiltonian schedule $\lambda(\tau/T)$ to improve the algorithm performance. We parametrize the Hamiltonian schedule as

$$\lambda\left(\frac{\tau}{T}\right) = \left(\frac{\tau}{T}\right) + \sum_{m=1}^{C} b_m \sin\left(\frac{m\pi\tau}{T}\right). \tag{4}$$

Here the schedule satisfies the boundary conditions $\lambda(0) = 0$ and $\lambda(1) = 1$, and the parametrization is complete if we take the high-frequency cutoff $C \to \infty$. Here, we set the cutoff C as 6. The parameters (b_m) are denoted by a vector $\mathbf{b}$ in the following. With $\mathbf{b} \to 0$, the schedule goes back to the linear form as used in standard AQC [16,17,19,35,36].

A. Reward settings

To perform the AQC Hamiltonian schedule configuration, we implement the soft actor-critic (SAC) method, a state-of-the-art RL algorithm dealing with continuous action control [37]. We have tested several different ways of reward settings to connect SAC with AQC, including Reward1, min[log*(success probability)],* which denotes the minimum of the logarithm of success probabilities, Reward2, ave*[*log*(success probability)]*, which denotes the average of the logarithm of success probabilities, Reward3, min*(success probability),* which denotes the minimum of success probabilities, Reward4, ave*(success probability)*, which denotes the average success probability, and Reward5, ave*(energy)*, which denotes the opposite of the average energy. The results with the five different reward settings are shown in Fig. 3. With the Reward1 setting, the configured AQC algorithm produces a relatively high mean value of success probability ∼0.145, which is larger than P_{th}, and the success probability distribution is narrowly distributed around the mean value. The Reward2 setting yields a larger mean value of success probability, but its distribution is too broad—there is a large number of problem instances having success probability below P_{th}. And the Reward3 setting leads to a similar performance as Reward1, with a mean success probability slightly smaller. The case with the Reward4 setting also produces a high mean value of success probability with a broader distribution, which is similar to the performance of the Reward2 setting. With

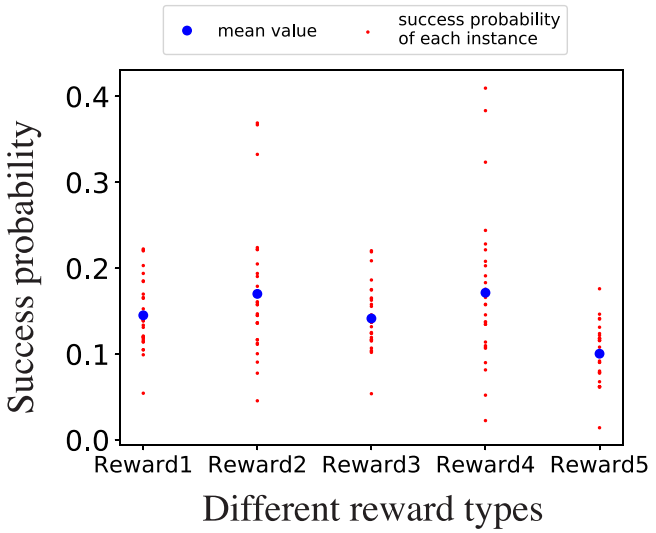

FIG. 3. Performance of SAC configured AQC with different reward settings. The success probabilities on all the hard factorization instances are obtained by using the best Hamiltonian schedule that is trained under the different reward settings within the same number of training steps. We test the different reward types including Reward1, min[log*(success probability)]*, Reward2, ave*[*log*(success probability)]*, Reward3, min*(success probability)*, Reward4, ave*(success probability)*, and Reward5, ave*(energy)*. The reward settings giving more weights to the worst-case problem instances such as Reward1 and Reward3 produce success probabilities more narrowly distributed. The averaging type of reward settings such as Reward2 and Reward4 yield a larger mean value of success probability but the distribution is much broader compared to Reward1 and Reward3. The energy type reward produces relatively lower success probability overall.

the Reward5 setting, although the success probability has a narrow distribution, the mean value is much lower than other reward settings. Through these numerical tests, we conclude that we need a proper type of reward signal to the SAC agent in the AQC configuration tasks to gain high success probability for all problem instances. It is a crucial prerequisite in the RL-based AQC configuration design scheme for the hard prime factorization instances. Conducting training processes under different reward settings leads to distinctive information flowing in the reinforcement learning scheme. The reward setting taking the success probability of the hardest instances (such as Reward1 and Reward3) provides a better guidance for the reinforcement learning method to configure the AQC algorithm. We choose the Reward1 setting in the following.

B. Training process

We apply SAC configuration on AQC-based prime factorization for a range of composite numbers from $N = 49$ to $N = 633$, whose Hamiltonian encoding has qubit numbers $n = 5, 6, 7, 8, 9, 10, 11$. We choose the AQC evolution time $T(n)$ according to the time when the average success probability of all the instances reaches the threshold ($P_{\text{th}} = 0.1$) in using the quadratic Hamiltonian schedule. From Fig. 4, it is evident that the reward in SAC converges for all qubit numbers within 1600 measurement steps. Taking the RL-designed

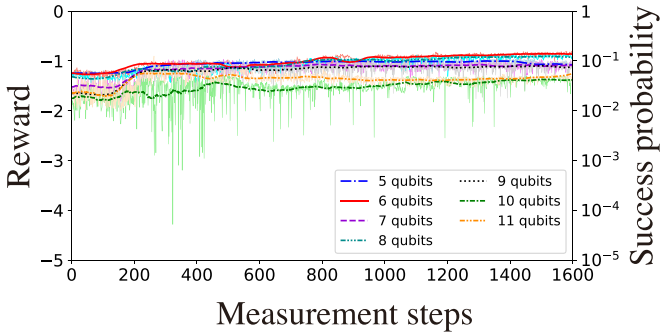

FIG. 4. Training process of SAC configuration on AQC-based prime factorization in the different system sizes with the qubit number of 5, 6, 7, 8, 9, 10, 11. We record the reward of min[log*(success probability)]*, which corresponds to the performance of the hardest factorizing instance in each system size during the training. The bold lines denote the results of smoothing over 80 nearby date points. We observe that the SAC improves the performance of the hardest instances and all the cases of rewards converge within 1600 measurement steps.

Hamiltonian schedule for AQC, the quantum factorization for all hard instances has a satisfactory success probability, roughly uniformly distributed. This results from our careful reward choice of using the minimum of the logarithm of success probability in training.

The slowing down problem caused by hard instances is thus rescued with our RL-configured AQC. We remark here that if a higher success probability ($P^\star$) is upon request, this can be achieved by simply repeating AQC multiple (M) times, with $M = [\log(1 - P^\star)]/[\log(1 - P_{\text{th}})]$ [17]. This relies on the fact that all the problem instances have a success probability above P_{th}. This repeating protocol does not necessarily work if only the average success probability reaches P_{th}.

C. Configured Hamiltonian schedules

We investigate the Hamiltonian schedule during the RL learning process for the prime factorization instances encoded by seven qubits. During the training of SAC, the reward signal converges within 1600 measurement steps (400 episodes with four measurements per episode) as shown in Fig. 4. We observe that the Hamiltonian schedule parameters $\mathbf{b}$ converge to a flattened plateau as shown in Fig. 5(a). The RL-configured Hamiltonian schedule is shown in Fig. 5(b). One characteristic feature of the RL-configured schedule is that it flattens at the beginning, and can be approximated by a power law $\lambda \sim (\tau/T)^7$.

The RL-configured Hamiltonian schedule strongly deviates from the unconfigured linear or quadratic schedule. The performances of different Hamiltonian schedules are shown in the inset of Fig. 5(b). The factorized integers are $N = 55, 65, 77, 91$, which are hard instances with unconfigured AQC. It is evident that the RL-configured Hamiltonian schedules exhibit substantially improved success probability—the success probability has doubled for the hard instances. The performance of the power-law schedule $\lambda \sim (\tau/T)^7$ in a seven-qubit system is better than linear and quadratic ones and is slightly poorer than the RL-configured one.

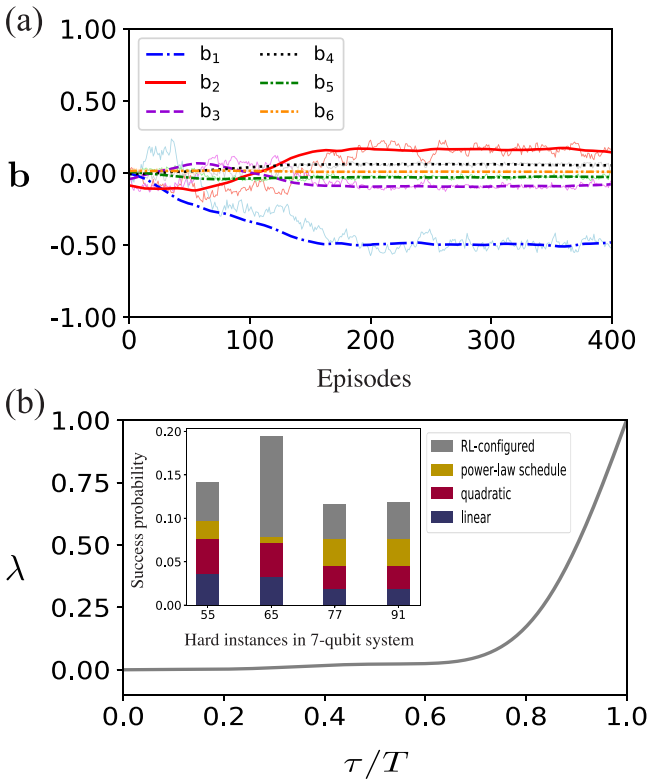

FIG. 5. Evolution of **b** during the SAC training process in the seven-qubit system. (a) The **b** values converge after 200 training episodes. For each episode, there are four measurement steps (see more details in Appendix B). (b) The representative RL-configured Hamiltonian schedule and the performance comparison of different schedules. We observe the RL-configured schedule has a large flattened region at the beginning. The corresponding performance of different schedules are shown in the inset. The success probabilities of the RL-configured schedule which is trained from a linear one are dramatically improved. The power-law schedule $\lambda \sim (\tau/T)^7$ has a slightly lower success probability compared with the RL-configured one.

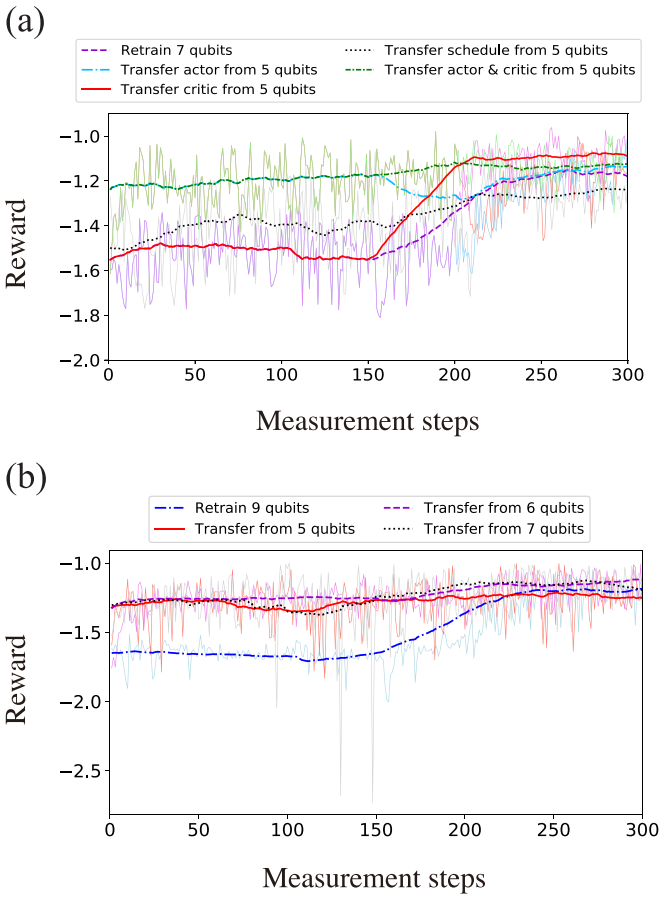

FIG. 6. Transferring methods in hard instance learning. The reward evolution during the learning process is shown in this plot. The bold lines denote the results of smoothing over 50 nearby date points. (a) Comparison of different transfer protocols on the seven-qubit system. We transfer the network data trained on five-qubit AQC to seven qubit here. The learning process of a direct retraining on the seven-qubit system is shown as a baseline ("purple dashed" line). The different protocols of transferring actor weights, critic weights, all network weights, and Hamiltonian schedule are presented by "blue dashed dotted," "red solid," "green dash dotted," and "black dotted" lines, respectively, in this plot. (b) Training curves of transferring the weights of actor and critic networks trained on five-, six-, and seven-qubit systems to a nine-qubit system. All the transfer learning processes reach an almost optimal reward within only a few measurement steps, much less than the direct training.

IV. TRANSFERABILITY OF THE REINFORCEMENT LEARNING IN AQC ALGORITHM CONFIGURATION

To further reduce the measurement cost, we investigate the transferability of our scheme. As the weights of learning networks are recorded during the training process, we test the different weight transferring methods, including Case I, transferring the recorded weights of actor networks, Case II, transferring the recorded weights of critic networks, and Case III, transferring both recorded weights of actor and critic networks. We also test the case of directly transferring the previous trained schedule as the initial ansatz in the new system size. We perform numerical tests by transferring from five-qubit AQC to seven-qubit. The results are shown in Fig. 6(a).

We find that the transfer protocol of Case III has a similar performance as Case I at the starting 150 measurement steps. After that, it is evident that Case III is more stable—it converges to optimal reward in a more systematic manner. The comparison between the transfer protocol of Case II and a direct training on a seven-qubit system is similar. Their collected rewards are very close to each other at the starting 150 measurement steps, with tiny difference unnoticeable in the plot. After that, the Case II transfer protocol is better than the direct retraining. Through these numerical tests, we conclude that the protocol of transferring actor and critic weights together outperforms other schemes in our AQC configuration task.

To test the robustness of the transfer learning protocol of taking all actor and critic weights, we further apply this protocol to a system of nine qubits. The results are shown in Fig. 6(b). We transfer the weights trained on AQC with five, six, and seven qubits to the nine-qubit system. These training processes taking the transfer data all have a faster convergence speed than a direct retraining on the nine-qubit system, confirming the robustness of the transfer learning protocol of taking all weights. The applicability of this type of transfer

learning for AQC configuration from smaller number of qubits to larger sizes also implies the scalability of our scheme.

V. CONCLUSION

We develop a reinforcement learning based scheme for adiabatic quantum algorithm configuration directly targeting the hard prime factorization instances. This is achieved by taking the minimum of the logarithm of the success probability of the AQC on factorization as the RL reward. By implementing the SAC algorithm, we find the learning process converges within only a few hundred measurements. Through numerical tests, we have shown that our developed AQC algorithm configuration scheme has fair transferability—the learning transferred from smaller qubit number systems to larger qubit numbers converges significantly faster than learning from scratch. This implies our AQC algorithm configuration scheme is potentially quite scalable.

ACKNOWLEDGMENTS

J.L. acknowledges helpful discussion with X. Zeng. This work is supported by the National Natural Science Foundation of China (Grants No. 11774067 and No. 11934002), the National Program on Key Basic Research Project of China (Grants No. 2017YFA0304204 and No. 2018YFB1305104), and the Shanghai Municipal Science and Technology Major Project (Grant No. 2019SHZDZX01), Shanghai Science Foundation (Grant No. 19ZR1471500).

APPENDIX A: ENCODING PROTOCOL: QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION (QUBO)

Here we provide the details of mapping the factorization problem to the quadratic unconstrained binary optimization (QUBO) problem. The problem of prime factorization is to factorize a given integer N into two integers p, and q, i.e., $N \to p \times q$.

For the first step, we divide the multiplication table (such as Table I) and calculate the total qubit number needed to factorize N. We define the bit length $L_p = \lfloor \log_2 p \rfloor$, $L_q = \lfloor \log_2 q \rfloor$, $L_N = \lfloor \log_2 N \rfloor$, as well as the bit string of $p : (p_{L_p} = 1, p_{L_p-1}, \ldots, p_1, p_0 = 1)_2$ and $q : (q_{L_q} = 1, q_{L_q-1}, \ldots, q_1, q_0 = 1)_2$, where $\lfloor a \rfloor$ denotes the largest integer not larger than a and assuming $p \leqslant q$ without loss of generality. We divide the multiplication table into B_K blocks with the width W, then $B_K = \lfloor (L_p + L_q)/W \rfloor$. We define ρ_i as the sum of the product terms in the block $i (i \geqslant 1)$:

$$\rho_i = \sum_{j=1+(i-1)\cdot W}^{i\cdot W} 2^{j-1-(i-1)\cdot W} \sum_{\substack{m+n=j \\ m\in[0,L_p] \\ n\in[0,L_q]}} p_m * q_n. \tag{A1}$$

We calculate the maximum value of carry variables and product terms in each block $i (i \geqslant 1)$,

$$\mathscr{B}_i = \mathscr{C}_{i-1} + \max \rho_i, \tag{A2}$$

where $\mathscr{C}_0 = 0$. To obtain $\mathscr{C}_i (i \geqslant 1)$, we define $\mathscr{M}_i = \lfloor \mathscr{B}_i / 2^W \rfloor$ and the number of carry variables c_i into block $i+1$,

$$c_i = \begin{cases} 0, & \text{if } \mathscr{M}_i = 0, \\ \lfloor \log_2(\mathscr{M}_i) \rfloor + 1, & \text{otherwise}, \end{cases} \tag{A3}$$

then $\mathscr{C}_i = 2^{c_i} - 1$. And we calculate $\chi_i = \sum_{k=1}^{i-1} c_k$, the total number of carry variables before (including) block i.

In the following, the total number of carry bits and summation terms in the cost function are defined as T_C and P_N, respectively. If $L_N - B_K \cdot W > 1$, then $T_C = \sum_{i=1}^{B_K} c_i$ and $P_N = B_K + 1$. Otherwise, $T_C = \sum_{i=1}^{B_K - 1} c_i$, $P_N = B_K$. The total auxiliary variable number T_A in reducing the higher-order coupling terms is $(L_p - 1) \cdot (L_q - 1)$. So the total qubit number required is

$$T_Q = (L_p - 1) + (L_q - 1) + T_C + T_A. \tag{A4}$$

Then we turn to the second step to construct the cost function. We define K_i as the sum of carry variable $\tilde{C}$ into block i,

$$K_i = \sum_{j=1}^{c_{i-1}} 2^{j-1} \tilde{C}_{\chi_{i-1}+j}, \tag{A5}$$

and F_i as the sum of the carry variable $\tilde{C}$ into block $i+1$,

$$F_i = \sum_{j=1}^{c_i} 2^{W+j-1} \tilde{C}_{\chi_i+j}, \tag{A6}$$

as well as V_i the target value of the block i which can be read out from the binary presentation of N directly.

We can write down the total cost function as

$$f_{\text{cost}} = \sum_{i=1}^{P_N} (\rho_i + K_i - F_i - V_i)^2, \tag{A7}$$

which can be expanded and simplified using $x^2 = x$ for $x = 0, 1$.

Then we turn to the third step to introduce auxiliary variables to reduce k-bit coupling terms ($k \geqslant 3$) using

$$\begin{aligned} x_1x_2x_3 &= \min_{x_4}[x_4x_3 + 2(x_1x_2 - 2x_1x_4 - 2x_2x_4 + 3x_4)], \\ -x_1x_2x_3 &= \min_{x_4}[-x_4x_3 + 2(x_1x_2 - 2x_1x_4 - 2x_2x_4 + 3x_4)], \end{aligned} \tag{A8}$$

TABLE I. Multiplication table for $143 = 11 \times 13$. The p and q rows are the binary representation of the prime factor and $\tilde{C}_i$ in the carries row is the carry bit. The bottom row is the binary representation of the number $N = 143$.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
p					1	p_2	p_1	1
q					1	q_2	q_1	1
					1	p_2	p_1	1
				q_1	p_2q_1	p_1q_1	q_1	
			q_2	p_2q_2	p_1q_2	q_2		
		1	p_2	p_1	1			
Carries			$\tilde{C}_2$	$\tilde{C}_1$				
$p \times q = 143$	1	0	0	0	1	1	1	1

by which the cost function is reduced to a quadratic form. The problem of prime factorization is then converted to the quadratic unconstrained binary optimization (QUBO) problem.

As the last step, we replace x_i as $(1-\sigma_{z_i})/2$, where σ_{z_i} is the Pauli Z matrix, and convert the cost function into the Ising type Hamiltonian H which would be suitable for AQC. As an illustration, we give a multiplication table for $143 = 11\times13$ in Table I.

The resulting equations derived from the two blocks are

$$\begin{aligned}&(1+p_2q_1+p_1q_2+1)\times 2^2\\&\quad+(p_2+p_1q_1+q_2)\times 2+(p_1+q_1)\\&=\tilde{C}_2\times 2^4+\tilde{C}_1\times 2^3+(111)_2\\&=16\tilde{C}_2+8\tilde{C}_1+7,\end{aligned}\tag{A9}$$

$$\begin{aligned}&1\times 2^2+(q_2+p_2+\tilde{C}_2)\times 2+(q_1+p_2q_2+p_1+\tilde{C}_1)\\&\quad=(1000)_2=8.\end{aligned}\tag{A10}$$

The corresponding cost functions are

$$\begin{aligned}f_1=&(4p_2q_1+4p_1q_2+2p_1q_1+2p_2+2q_2\\&+p_1+q_1-16\tilde{C}_2-8\tilde{C}_1+1)^2,\end{aligned}\tag{A11}$$

$$f_2=(p_2q_2+2p_2+p_1+2q_2+q_1+2\tilde{C}_2+\tilde{C}_1-4)^2.\tag{A12}$$

The high-order $k(k\geqslant 3)$ terms in the cost function can be reduced using Eq. (A8). As mentioned before, with the variable replacement in the last step, we can get the Ising type Hamiltonian.

APPENDIX B: FORMULATING ADIABATIC ALGORITHM DESIGN AS A MARKOV DECISION PROCESS

Aiming at an automated architecture for optimal design of quantum adiabatic factorization, we use a Markov decision process (MDP) [38] for the configuration of the **b**-parametrized Hamiltonian schedule. In MDP, we update the vector **b** according to a stochastic policy π, which generates a sequence $\mathbf{b}_{t+1}=\mathbf{b}_t+\mathbf{a}_t$, with $\mathbf{a}_t$ random variables drawn according to π. In our study, we consider a MDP with a finite depth ($L_{\mathbf{M}}$). The optimal quantum adiabatic algorithm design is then converted to searching for a MDP policy converging to a **b** vector that maximizes the success probability of AQC.

Quantitatively, the optimal policy is defined to be one that maximizes an objective of long-term reward $\mathbf{E}_\pi[\sum_t(\gamma^t r_t)]$, with $\gamma\in(0,1]$ a discount factor and r_t a reward at tth step. The reward is defined by the success probability of AQC taking the configuration $\mathbf{b}_t$.

We compare the performances of different reward settings and choose min[log(*success probability*)] as the reward. This choice automatically gives extra weight to the most difficult problem instances having low success probability.

There are multiple RL protocols to specify the MDP policy. Here we focus on the entropy-regularized soft actor-critic (SAC) algorithm [37]. For SAC, the objective return is modified to the form of $\mathbf{E}_\pi[\sum_t(\gamma^t r_t+\alpha\mathcal{H}[\pi(\cdot|s_t)])]$, where $\mathcal{H}$ is the causal policy entropy and α determines its relative weight. The modification balances the trade-off between exploration and exploitation. The state of RL agent is specified to be $\mathbf{s}_t=\{\text{DoubleOneHotEncoder}(t),\mathbf{b}_t\}$, with more details of problem setting discussed in Appendix B 2. At each iteration, the agent chooses an action $\mathbf{a}_t$ by sampling from the actor network and acts (adds) to $\mathbf{b}_t$. We assume $\mathbf{b}_t\in[-1,1]$ and set the action element $a_m\in[-0.01,0.01]$. In SAC training, the actor is supervised by the critic, who processes the reward signal and gets updated according to the temporal-difference error using Polyak averaging [39] with a target to stabilize training. The whole architecture for our quantum algorithm configuration is illustrated in Fig. 1. More details are provided in the following parts.

1. Pseudocode of reinforcement learning process

Algorithm 1. SAC-configured AQC

Require X: prime factorization instances; T: evolution time; θ: initialized policy parameters; ϕ, $\hat{\phi}$: initialized critic and target parameters; $\mathbf{b}_0$: initialized schedule; Δt_w: measure frequency; P_{th}: success probability threshold
Ensure configured schedule $b_1, b_2, \ldots, b_C$
  $\mathbf{H}_{\text{p}} \longleftarrow$ QUBOEncoding(**X**)
  $\mathbf{X}_{\text{hard}} \longleftarrow$ SELECT(**X**, P_{th})
  $\mathbf{b} \leftarrow \mathbf{b}_0$
  for each episode **do**
    for each step t in episode length **do**
      $\mathbf{s}_t \leftarrow$ [DoubleOneHot(t), $\mathbf{b}_t$]
      Sample $\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t)$
      $\mathbf{b}_{t+1} \leftarrow \mathbf{b}_t + \mathbf{a}_t$
      while MonotonicConstraint($\mathbf{b}_{t+1}$) is false **do**
        Resample $\mathbf{a}_t \sim \pi_\theta(\mathbf{s}_t)$
        $\mathbf{b}_{t+1} \leftarrow \mathbf{b}_t + \mathbf{a}_t$
      end while
      $\mathbf{s}_{t+1} \leftarrow$ [DoubleOneHot($t+1$), $\mathbf{b}_{t+1}$]
      If t mod $\Delta t_w = 0$ **then**
        $r_t \leftarrow$ QuantumEvolution($\mathbf{b}_{t+1}$, T, $\mathbf{X}_{\text{hard}}$)
      else
        $r_t \leftarrow 0$
      end if
    end for
    Buffer $\leftarrow (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$
    for each gradient step **do**
      $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}\text{oss}(\theta)$
      $\phi \leftarrow \phi - \nabla_\phi \mathcal{L}\text{oss}(\phi, \hat{\phi})$
      $\hat{\phi} \leftarrow \eta\hat{\phi} + (1-\eta)\phi$
    end for
  end for

2. DoubleOneHot environment

The episode length, quantum schedule, and time step in the episode are denoted by L, $\mathbf{b}\in\mathcal{R}^6$, and $t\in[1,L]$, respectively.

State. State s_t contains a sequential encoder, in which we use two one-hot encoders with 10 bits, respectively, representing 100 steps at large with fixed 20 dimensions. Hence this sequential encoder turns episode length L into a one-hot vector with 20 dimensions and six parameters in Hamiltonian schedule are denoted as $\mathbf{b}\in\mathcal{R}^6$.

Action. The learned policy maps state into action denoted as $\mathbf{a}\in\mathcal{R}^6$. At each step t, we define action $\mathbf{a}_t\in\mathcal{R}^6$ as how

far should the agent reach based on current schedule $\mathbf{b}_t \in \mathcal{R}^6$ and current step encoded by two one-hot encoders denoted as double-one-hot(t).

Reward. Measure the action $\mathbf{a}_t$ sampled from policy π_θ if it satisfies the monotonic constraint enforced on the evolution path. And return the minimum logarithm of success probability as a reward signal.

DoubleOneHotEnv. The problem setting can be abbreviated as follows:

$$\mathbf{s}_t = [\text{double-one-hot}(t), \mathbf{b}_t] \in \mathcal{R}^{26}, t \in \{1, 2, \ldots, 100\},$$
$$\pi : \mathbf{a}_t \sim \pi_\theta(\cdot|s_t),$$
$$r_t = \text{Measure}(\mathbf{a}_t + \mathbf{b}_t \triangleq \mathbf{b}_{t+1}),$$
$$\mathbf{s}_{t+1} = [\text{double-one-hot}(t+1), \mathbf{b}_{t+1}] \in \mathcal{R}^{26}.$$

3. Hyperparameters

Hyperparameters used in the SAC-configured AQC algorithm are listed below, which are categorized into *Env*, *Learner*, and *Network*. In terms of *Env* hyperparameters, we specify the parameter space in Hamiltonian schedule and action stride at first and then formulate the state and Markov decision process as known environment setting. Subsequently, the exploration space is mainly determined by episode length. The reward signal is measured once after n steps and amplified by reward scale. Finally, other direct hyperparameters about *Learner* and *Network* are listed in Table II.

TABLE II. SAC-configured AQC parameters.

Parameter	Value
Env	
Action stride	$[-0.01, 0.01]$
Schedule parameter space	$[-1.0, 1.0]$
Environment setting	DoubleOneHotEnv
Number of episodes	1000
Episode length	40
Reward scale	5
Measure every n steps	10
Reward type	min[log(success probability)]
Learner	
Optimizer	Adam
Learning rate	3×10^{-4}
Discount	0.9
Alpha	0.02
Polyak (η)	0.995
Target update interval	2
Gradient steps	2
Network	
Number of hidden layers	2
Actor hidden layer size	256
Critic hidden layer size	512
Batch size	128
Actor maximum std	1
Actor minimum std	-10
Random steps	0
Buffer size	10^6

[1] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics* (Springer, New York, 2010), Vol. 2.

[2] G. Meletiou, D. Tasoulis, and M. N. Vrahatis, in *Proceedings of the IASTED 2002 Conference on Artificial Intelligence* (IASTED, Banff, Canada, 2002), pp. 483–488.

[3] A. Stekel, M. Shukrun, and A. Azaria, in *Proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (IEEE, New York, 2018), pp. 502–507.

[4] R. V. Yampolskiy, Int. J. Bio-Insp. Comput. **2**, 115 (2010).

[5] J. V. Monaco and M. M. Vindiola, in *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, New York, 2017), pp. 1–4.

[6] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, Nature (London) **573**, 390 (2019).

[7] P. W. Shor, SIAM Rev. **41**, 303 (1999).

[8] J. Preskill, Quantum **2**, 79 (2018).

[9] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature (London) **414**, 883 (2001).

[10] A. Dash, D. Sarmah, B. K. Behera, and P. K. Panigrahi, arXiv:1805.10478.

[11] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, Science **292**, 472 (2001).

[12] X. Peng, Z. Liao, N. Xu, G. Qin, X. Zhou, D. Suter, and J. Du, Phys. Rev. Lett. **101**, 220405 (2008).

[13] N. Xu, J. Zhu, D. Lu, X. Zhou, X. Peng, and J. Du, Phys. Rev. Lett. **109**, 269902 (2012).

[14] S. Jiang, K. A. Britt, A. McCaskey, T. Humble, and S. Kais, Sci. Rep. **8**, 17667 (2018).

[15] R. Dridi and H. Alghassi, Sci. Rep. **7**, 1 (2017).

[16] K. Xu, T. Xie, Z. Li, X. Xu, M. Wang, X. Ye, F. Kong, J. Geng, C. Duan, F. Shi, and J. Du, Phys. Rev. Lett. **118**, 130504 (2017).

[17] T. Albash and D. A. Lidar, Rev. Mod. Phys. **90**, 015002 (2018).

[18] B. Wang, F. Hu, H. Yao, and C. Wang, Sci. Rep. **10**, 1 (2020).

[19] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, Rep. Prog. Phys. **83**, 054401 (2020).

[20] I. H. Deutsch, PRX Quantum **1**, 020101 (2020).

[21] M. Mosca and S. R. Verschoor, Sci. Rep. **12**, 7982 (2022).

[22] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Phys. Rev. X **8**, 031086 (2018).

[23] M. M. Wauters, E. Panizon, G. B. Mbeng, and G. E. Santoro, Phys. Rev. Research **2**, 033446 (2020).

[24] J. Lin, Z. Y. Lai, and X. Li, Phys. Rev. A **101**, 052327 (2020).

[25] E. Boros, P. L. Hammer, and G. Tavares, J. Heuristics **13**, 99 (2007).

[26] M. Lewis and F. Glover, Networks **70**, 79 (2017).

[27] R. Patton, C. Schuman, and T. Potok, Quantum Inf. Process. **18**, 117 (2019).

[28] W. Peng, B. Wang, F. Hu, Y. Wang, X. Fang, X. Chen, and C. Wang, Sci. China Phys. Mech. Astron. **62**, 60311 (2019).

[29] V. Bapst, L. Foini, F. Krzakala, G. Semerjian, and F. Zamponi, Phys. Rep. **523**, 127 (2013).

[30] J. Roland and N. J. Cerf, Phys. Rev. A **65**, 042308 (2002).

[31] A. T. Rezakhani, A. K. Pimachev, and D. A. Lidar, Phys. Rev. A **82**, 052305 (2010).

[32] H. Hu and B. Wu, Phys. Rev. A **93**, 012345 (2016).
[33] L. Zeng, J. Zhang, and M. Sarovar, J. Phys. A: Math. Theor. **49**, 165305 (2016).
[34] B. Irsigler and T. Grass, Quantum **6**, 624 (2022).
[35] X. Qiu, P. Zoller, and X. Li, PRX Quantum **1**, 020311 (2020).
[36] A. Saxena, A. Shukla, and A. Pathak, Quantum Inf. Process. **20**, 112 (2021).
[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, in *International Conference on Machine Learning* (PMLR, Stockholm, Sweden, 2018), pp. 1861–1870.
[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).
[39] B. T. Polyak and A. B. Juditsky, SIAM J. Control Optim. **30**, 838 (1992).