Dynamic Control of Self-assembly of Quasicrystalline Structures through Reinforcement Learning SUPPLEMENTARY INFORMATION

Uyen Tu Lieu*[a,b] and Natsuhiko Yoshinaga[a,b]

[a] *Future University Hakodate, Kamedanakano-cho 116-2, Hokkaido 041-8655, Japan*
[b] *Mathematics for Advanced Materials-OIL, AIST, 2-1-1 Katahira, Aoba, 980-8577 Sendai, Japan*
E-mail: uyenlieu@fun.ac.jp; yoshinaga@fun.ac.jp

S1 Reinforcement learning and Q-learning

The basic ingredients of reinforcement learning (RL) include an agent, an environment, and reward signals. The agent observes the states s of the environment and learns optimal actions a through a policy π that maximises the cumulative future rewards R [1, 2, 3]. The future reward is the sum of the instantaneous reward r_i at each step i

$$R = \sum_i \gamma r_i, \tag{S1}$$

with the discount factor γ. Formally, RL is expressed by a tuple of $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, g, \pi\}$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P}$ is a Markov transition process of the environment describing its time evolution, g is the (instantaneous) reward function, and π is a policy. The transition process $P(s_{t+1}|s_t, a_t) \in \mathcal{P}$ maps the current state $s_t \in \mathcal{S}$ to the next state s_{t+1} under the action $a_t \in \mathcal{A}$. The process is supplemented by the initial probability of the states $P(s_0)$. The reward measures whether the current state is good or bad. The reward function gives some numbers from the current state and action as $r_t = g(s_t, a_t)$. In this work, we assume the reward function is dependent only on the state, that is, $r_t = g(s_t)$. In general, the policy is a conditional probability $\pi(a|s)$ of taking the action under a given state. We assume the deterministic policy $a(s)$, namely, the action is the function of the state.

A physical interpretation of RL is to estimate the best dynamic control strategy to get a desired structure or physical property. The physical system of variables s_t yields the dynamics expressed by the Markov process $\mathcal{P}(s_{t+1}|s_t, a_t)$ under an external force and/or parameter change in the model expressed by a_t. At each time, we can compare the current state s_t and the target state s^*. The distance between them is an instantaneous reward. The goal of RL is to estimate the best policy from which we choose the action a as a function of the current state s.

There are many RL algorithms to train the agent. Q-learning is a popular algorithm for learning optimal policies in Markov decision processes [2]. It is a model-free, value-based algorithm that uses the concept of Q-values (Quality value) to guide the agent's decision-making process. Q-value, denoted as $Q(s, a)$, is the cumulative reward obtained by taking action a on the current state s and then following the optimal policy. The simplest Q-learning uses a Q-table in which Q-values are updated at each point in discretised action and state spaces. The size of Q-table depends on the number of elements of the state spaces and action spaces. For example, consider a system with two state spaces discretised into m and n elements, and an action space with l elements. In this case, the corresponding Q-table is a three-dimensional array with dimensions $m \times n \times l$. This array represents the whole state-action space, in which the agent (we) can store and update Q-values for all possible combinations of states and actions. The policy is then extracted from the Q-value of each state-action pair $Q(s, a)$. In general, the algorithm involves many epochs (or episodes). The Q-table is initialised at first. For each epoch, the states are also initialised, then for each step in the epoch, we perform the following algorithms:

- Observe a current state $s_t \in \mathcal{S}$.
- Select and perform an action $a_t \in \mathcal{A}$ based on the policy from $Q(s, a)$.

- Observe the subsequent state s_{t+1}.
- Receive an immediate reward r_{t+1}.
- Update iteratively the Q-function by

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)], \tag{S2}$$

where the learning rate α is a hyperparameter $0 \leq \alpha \leq 1$ that reflects the magnitude of the change to $Q(s_t, a_t)$ and the extent that the new information overrides the old information. If $\alpha = 0$, no update at all; if $\alpha = 1$, then completely new information is updated in Q. The discount factor γ is associated with future uncertainty or the importance of the future rewards ($0 \leq \gamma \leq 1$).

In RL, it is important to consider the balance between exploitation and exploration. If we just follow the current (non-optimal) policy, it is unlikely to find potentially more desired states. On the other hand, if our search is merely random, it takes a significant amount of time to find them. In Q-learning, exploitation involves selecting the action that is believed to be optimal, i.e. maximum Q-value, while exploration involves selecting the action that does not need to be optimal within the current knowledge. To balance these strategies, the ϵ-greedy method is used. In the ϵ-greedy method, a random action at each time step is selected with a fixed probability $0 \leq \epsilon \leq 1$ instead of the optimal action with respect to the Q-table.

$$\pi(s) = \begin{cases} \text{random action } a \in \mathcal{A}, & \text{if } \xi < \epsilon \\ \arg\max_{a \in \mathcal{A}} Q(s, a), & \text{otherwise,} \end{cases} \tag{S3}$$

where $0 \leq \xi \leq 1$ is a uniform random number at each step.

S2 Reinforcement learning with value iteration method

Q-learning is a model-free method in RL. As shown in the main article, the Q-table is updated during training (Brownian dynamics simulations at given T) and eventually the policy is determined from the Q-table. Here we propose to use value iteration to utilise the data from training the Q-table. Value iteration is an important method using dynamic programming to find the optimal policy. It is a model-based method, i.e. we need to know the model dynamics (transition probability of the next states given current states and actions) [1, 2, 3]. Therefore, we first estimate the transition probability from the current state $s = (\sigma, T)$ to the next state s' under the action a, $P(\sigma', T'|\sigma, T, a)$, from empirical sampling. Then, we use Bellman's equation to estimate the value function, from which we can estimate the policy of the temperature change. Here, we show how to calculate the value function

1. Sampling data $S(s, a, s')$
2. Discretising the state spaces and calculating the transition probability $P(s'|s, a)$
3. Performing value iteration on the discretised state space:
   - initialise the value function $V(s) = 0$
   - in each iteration, calculate

$$Q(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma V(s')]; \tag{S4}$$

and the value function in this iteration is $V(s) = \max_a Q(s, a)$.

For the self-assembly of patchy particles in our study, there are two states $s = (\sigma, T)$. The transition probability is estimated from the trained datasets shown in Table S4, corresponding to 72,000 samples. We report the result after 100 iterations when the value function converges, i.e. the ratio of flipped policy reaches zero (Fig. S1). Note that the calculation of $V(\sigma, T)$ and policy uses the information of $P(\sigma', T'|\sigma, T, a)$. The hyperparameters such as the target σ^*, reward function, can also be varied. Once the value function converges, one can determine the corresponding Q-value $Q(\sigma, T, a)$ and the policy $\pi(a|\sigma, T) = \arg\max_a Q(\sigma, T, a)$. The reward function and discount factor γ in value iteration are chosen identical to that in Q-learning.

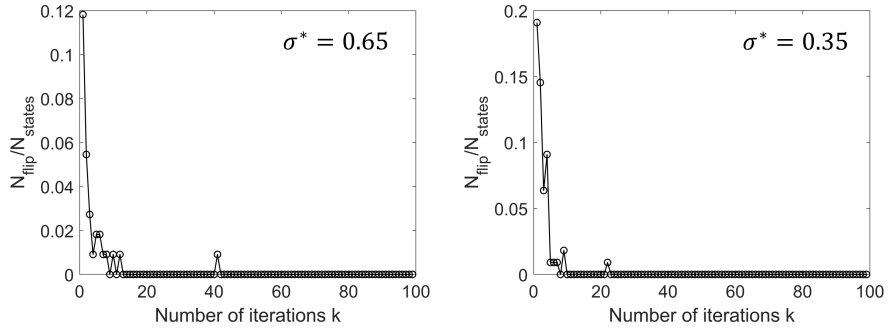

Figure S1: Convergence of the policy by value iteration method.

S3 Replica Exchange Monte Carlo simulations

Replica Exchange Monte Carlo method (REMC, also called parallel tempering) [4, 5] is employed to investigate the equilibrium phase diagram of the five-fold patchy particles used in the study. In REMC, $N_{\rm rep}$ noninteracting replicas are simulated simultaneously at a range of inverse temperature $\beta_1 \leq \beta_2 \leq ... \leq \beta_{N_{\rm rep}}$ by standard Monte Carlo (MC) simulations [6]. After a fixed number of Monte Carlo sweeps, extra replica exchanges (swapping) between replicas of neighbouring parameter β_i and β_{i+1} is suggested and accepted with a probability

$$p(E_i, \beta_i \to E_{i+1}, \beta_{i+1}) = \min(1, \exp((\beta_i - \beta_{i+1})(E(\mathbf{x}_i) - E(\mathbf{x}_{i+1})), \tag{S5}$$

where $\mathbf{x}$ is the particle configuration.

In the study, we set the distribution of inverse temperature $\{\beta_i\}$ on replicas based on the distribution of energy $E(\mathbf{x}, \beta)$. In order to facilitate the exchange, we chose β so that $(\beta_i - \beta_{i+1})(E(\mathbf{x}_i) - E(\mathbf{x}_{i+1}) \sim \mathcal{O}(1)$. We use the same initial condition for all replicas. Initially, the configuration of each replica evolves under its temperature. When exchange occurs at each REMC step, the temperatures between the neighbouring temperatures are swapped, i.e. $(\beta_i, \beta_{i+1}) \to (\beta_{i+1}, \beta_i)$. The simulation condition for REMC simulation is given in Table S1.

Parameter	Value
Potential	Patchy particle
Number of particles, N	256
Area fraction	0.75
Initial configuration	DDQC structure
Number of replicas, $N_{\rm rep}$	44
Temperature	$[0.2, 3.0]$
Number of MC sweeps per replica exchange (RE) step	100,000
Number of replica exchange steps	1,000

Table S1: Setting of REMC simulations. Details of temperatures can be seen in Fig. S2-S3.

State	σ	Z	H
σ phase	$\sigma \geq 0.93$	$Z < 0.02$	
DDQC	$0.7 \leq \sigma < 0.93$	$0.04 \leq Z \leq 0.12$	
Z-phase	$\sigma < 0.2$	$Z \geq 0.2$	
Fluid	$\sigma < 0.2$	$Z < 0.2$	$H < 0.2$

Table S2: Criteria for the structures in REMC. The Fourier transformations of the DDQC, Z-phase snapshots have 12-fold symmetric spots, triangulated spots, respectively. When the assembly does not belong to any state, we name it as 'other'.

The profile of the temperature of each replica index during REMC simulations is given in Fig. S2. The energy corresponding to the temperature in Fig. S3 shows a huge gap at $T > 0.9$ and $T < 0.85$, indicating a change of phase. We determine the ratio of the local structure σ, Z, H, U (undefined) at each REMC step. The probability distributions of those ratios are depicted in Fig. S4. In order to define the state of the self-assembly, e.g. dodecagonal quasicrystal (DDQC), σ phase, etc., we use the criteria based on the local structures (Table S2). Figure S5 illustrates the probability distribution of these states as a function of temperature. We assume that the state

with the highest probability is the dominant state. There are three main phases: DDQC, Z-phase and fluid. The change between Z-phase and DDQC corresponds to the huge energy gap in Fig. S3. For the temperature $T \in [0.2, 1.3]$ used in the reinforcement learning, the phases are DDQC and Z-rich. The dependence of the local structure on temperature with the phase is presented in the main text.

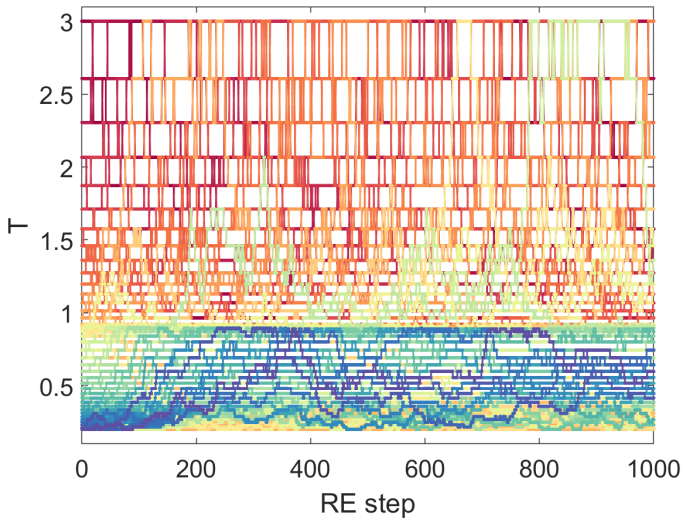

Figure S2: Temperature profiles of replica exchange Monte Carlo simulations for the system of $N = 256$, $A = 0.75$ (Table S1). Colour corresponds to the replica index.

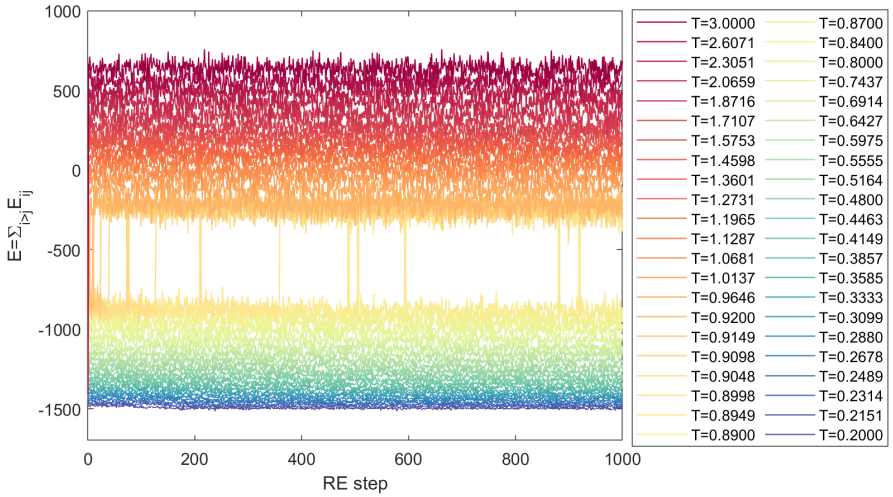

Figure S3: Energy profiles for the replica exchange system in Table S1. Colour corresponds to temperature.

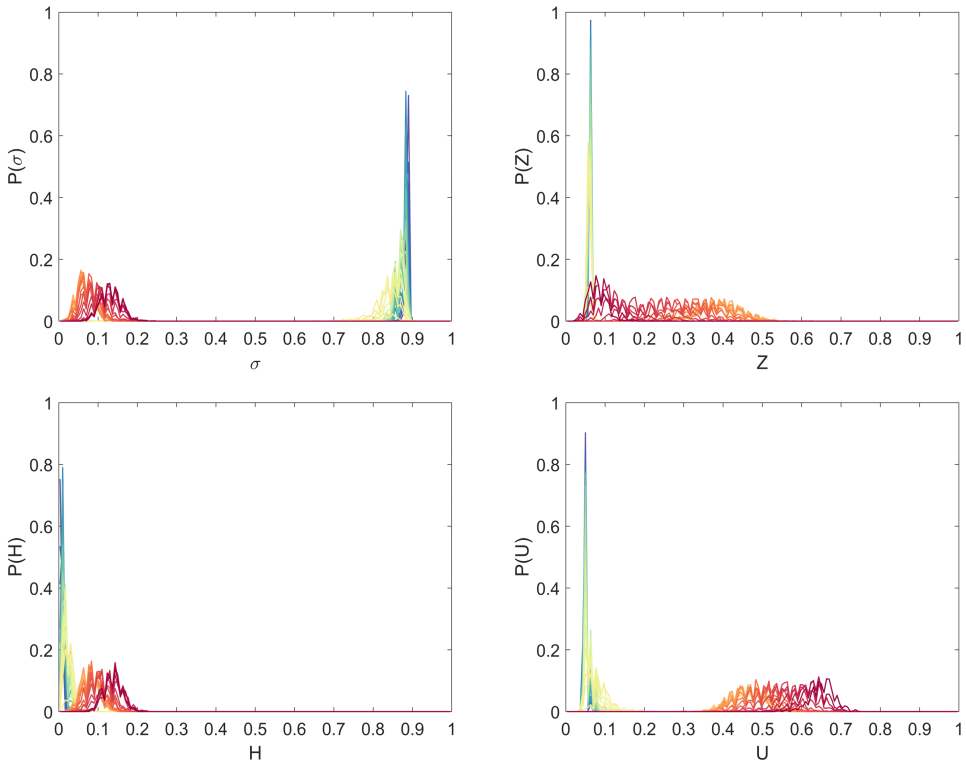

Figure S4: Probability distributions of the ratios of local structure for the replica exchange system in Table S1. The data is taken from RL step 200 to 1000. Colour corresponds to temperature like Fig. S3.

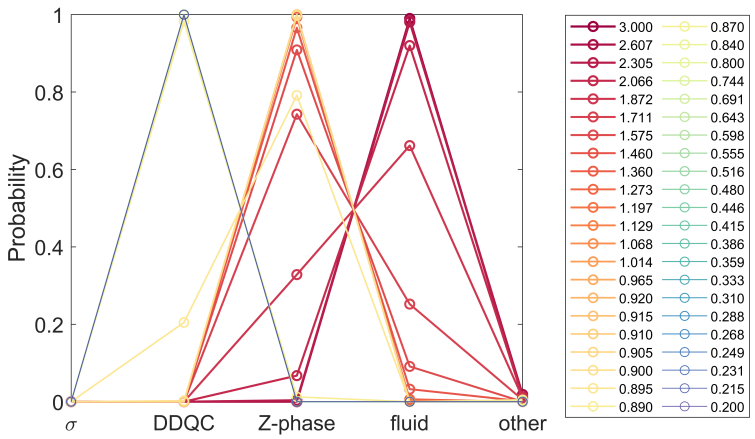

Figure S5: Probability distributions of the local structure ratios for the replica exchange system in Table S1. Colour corresponds to temperature like Fig. S3. The unknown state 'other' occupies only a small portion, thus brings no major effect to the phase.

S4 Quenching simulations at fixed temperatures

Brownian dynamics simulations of patchy particles at fixed temperature setting are performed at the conditions similar to RL test. The initial particle configurations are randomly assigned. Temperature is fixed during simulation. The number of particles is $N = 256$. The spherical particles are confined in a flat plane of the size $L_x \times L_y \times 2a$ where a is the particle radius and $L_x = L_y$. The periodic boundary condition is applied on L_x and L_y. The area fraction is defined as $A = \pi a^2 N/(L_x L_y) = 0.75$. The number of Brownian steps is 40×10^6. The dimensionless time step is set as $\Delta t = 0.0005T$ when $T < 2$, and $\Delta t = 0.0001T$ when $T \geq 2$. The details of the local structure σ, Z, H, U (undefined) at the end of simulations are given in the Fig. S6.

We investigate the behaviour of orientation of the particles. As depicted in Fig. S7, the ori-

entation of the patchy particle i is defined by an orthogonal basis $\hat{\mathbf{n}}_m^{(i)}$, $m = 1, 2, 3$, where $\hat{\mathbf{n}}_3^{(i)}$ corresponds to the basis vector of uniaxial symmetry and $\hat{\mathbf{n}}_1^{(i)}$ and $\hat{\mathbf{n}}_2^{(i)}$ lie in the equatorial plane in the framework of spherical harmonics. For the patchy particle Y_{55} in our study, the centres of particles are set in xy plane and the particles can rotate freely. In each snapshot, we calculate the property $\langle \hat{\mathbf{n}}_3 \rangle_z = \sum_i^N |\hat{\mathbf{n}}_{3z}^{(i)}|/N$, which expresses how much the plane of five patches deviates from the xy plane. The dependence of the orientation on temperature in quenching setting is presented in Fig. S7. The samples in this figure are the same as in Fig. S6. The result suggests that the uniaxial axis $\langle \hat{\mathbf{n}}_3 \rangle$ is perpendicular to the xy plane for DDQC structure, while it becomes more random as the temperature increases. It suggests that the out-of-plane rotations are rare in DDQC structures, but they occur more frequent in the Z-rich phase.

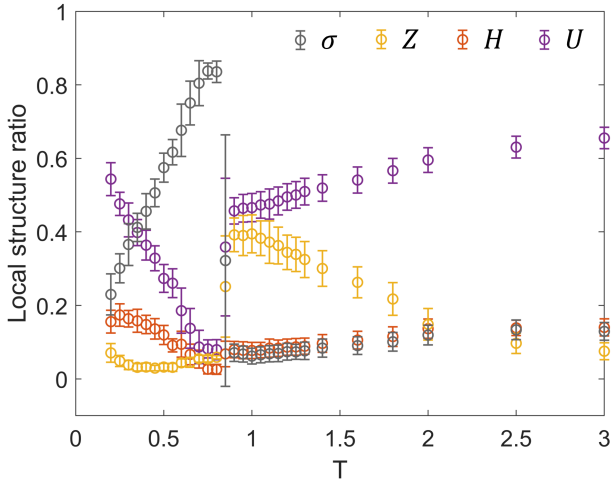

Figure S6: Assemblies by Brownian dynamics simulation at fixed temperature. The mean (circle) and standard deviation (bar) of the local structures are calculated over 10 independent samples. We use 20 snapshots at the end of each sample.

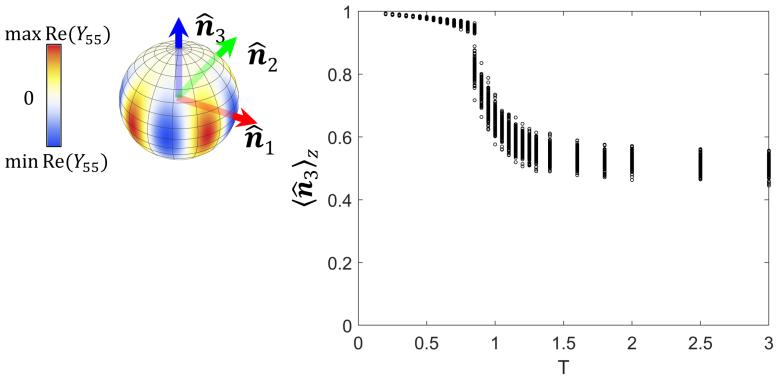

Figure S7: Orientational property of the assemblies by Brownian dynamics simulation at fixed temperature in Fig. S6. Illustration of the patchy particle and its basis are given.

S5 Details of local structure ratios

The details of local structure ratios for patchy particle assemblies at different temperature settings are given in Table S3.

	Simulation conditions	σ	H	Z	U	Remarks
1	$N = 256$, A=0.75, RL for DDQC	0.781 ± 0.041	0.058 ± 0.023	0.052 ± 0.015	0.109 ± 0.035	DDQC
2	$N = 256$, A=0.75, annealing	0.761 ± 0.070	0.068 ± 0.030	0.055 ± 0.009	0.116 ± 0.057	DDQC
3	$N = 256$, A=0.75, T=0.6	0.637 ± 0.066	0.106 ± 0.023	0.037 ± 0.011	0.220 ± 0.061	metastable
4	$N = 256$, A=0.75, T=0.7	0.793 ± 0.062	0.049 ± 0.027	0.052 ± 0.012	0.106 ± 0.051	DDQC
5	$N = 1024$, A=0.75, annealing	0.812 ± 0.031	0.044 ± 0.011	0.061 ± 0.003	0.076 ± 0.026	DDQC
6	$N = 1024$, A=0.78, annealing (Ref.[7])	0.729 ± 0.030	0.076 ± 0.011	0.069 ± 0.007	0.126 ± 0.024	DDQC
7	$N = 256$, A=0.75, REMC at T=0.87	0.842 ± 0.023	0.024 ± 0.011	0.059 ± 0.004	0.075 ± 0.021	DDQC

Table S3: Ratio of local structures of assemblies obtained by various temperature settings: reinforcement learning, annealing, and quenching, and REMC. The temperature in annealing is shown in the main text. There are at least 10 independent samples for the calculation of mean and standard deviation.

S6 Testing policy for DDQC target at different initial structures

Here we investigate the performance of the policy obtained by Q-learning for 101 epochs with DDQC target $\sigma^* = 0.91$ (see Fig. 4 in main text). This policy is trained with the condition that the initial structure is randomly assigned ($\sigma_0 < 0.2$). The policy covers quite wide range of σ and T. We used two different initial conditions $\sigma_0 = 0.56$ and $\sigma_0 = 0.87$ and performed the test using the estimated policy with $\sigma^* = 0.91$. The results are shown in Fig. S8. The mean and standard deviation of 20 independent tests are 0.74 ± 0.06 for $\sigma_0 = 0.56$ and 0.84 ± 0.03 for $\sigma_0 = 0.87$. The estimated policy works well for $\sigma_0 = 0.87$, but for the initial structure with $\sigma_0 = 0.56$, some tests work well while some do not. This may be because the small T and intermediate σ region in the estimated policy was not trained well. In fact, the region is still converging at 101 epochs, as we can see from Fig.4(e).

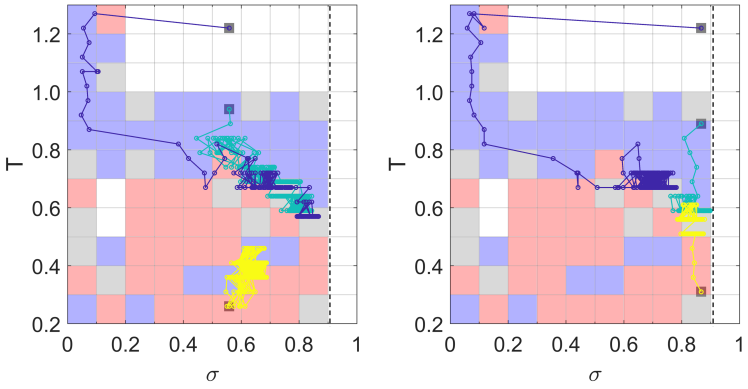

Figure S8: Performance of the policy at different initial particle configurations (left) $\sigma_0 = 0.56$ and (right) $\sigma_0 = 0.87$. Selected tests on the policy obtained in Fig. 4 are given. The initial points are marked with filled square. The mean and standard deviation of 20 independent tests for $\sigma_0 = 0.56$ and $\sigma_0 = 0.87$ are 0.74 ± 0.06 and 0.84 ± 0.03, respectively. The dashed line is located where $\sigma = \sigma^* = 0.91$.

S7 Reinforcement learning for DDQC of isotropically interacting particles

In this section, we perform RL for target DDQC assembled by particles interacting with the isotropic potential. The purpose is to show the versatility of RL in handling different physical systems. The training uses Q-learning with 101 epochs, and other hyperparameters are the same as those for RL with patchy particles, except that the number of steps in each epoch is 100, and the area fraction is 0.81. This is because the isotropic system self-organises faster, and has distance to the nearest neighbour shorter than patchy particle system does.

As shown in Fig. S9(a), the trained policy for the DDQC target from particles with isotropic interactions has many 'blue' and 'grey' elements, suggesting that the target DDQC can be obtained

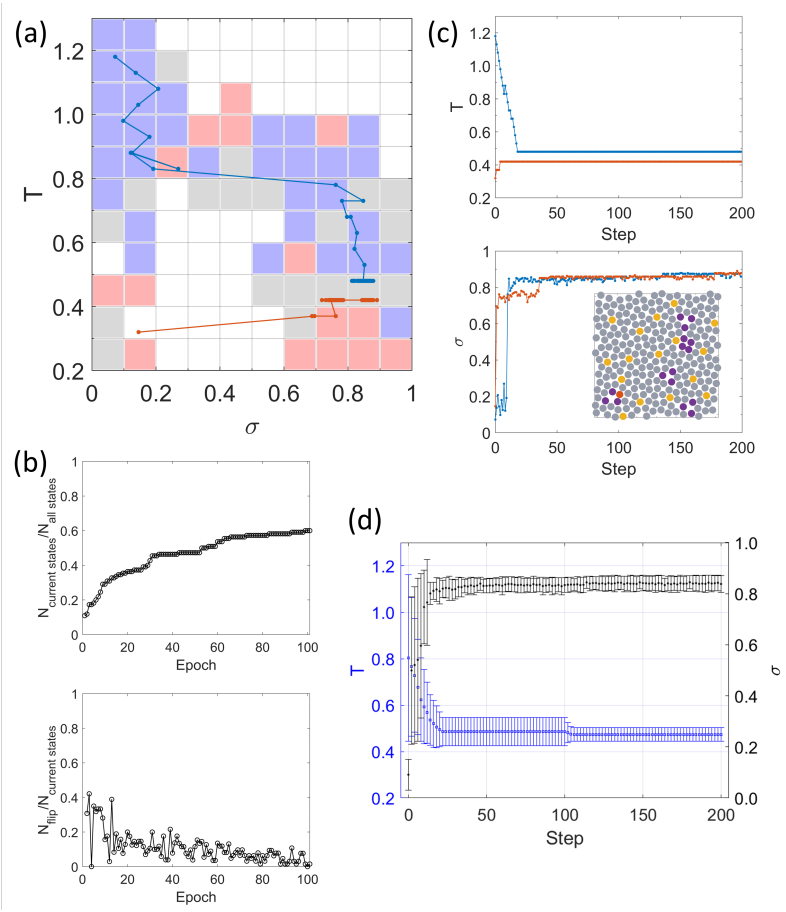

Figure S9: Reinforcement learning for DDQC of isotropically interacting particles. (a) The trained policy with trajectories of selected tests. (b) Convergence of the policy during training. (c) Corresponding temperature and σ of the tests in (a). The snapshot at the last point of the red trajectory is given. (d) Mean and standard deviation of temperature (blue) and σ (black) from 20 independent samples.

by decreasing the temperature low enough. The existence of a characteristic temperature similar to that of patchy particle system (Fig. 4-5) is not clearly observed. Although the policy slowly converges during training (Fig. S9(b)), the policy still works. In fact, the two representative tests in Fig. S9(a,c) confirm that DDQC can be formed by a simple policy. For the test starting from high initial temperature $T_0 \approx 1.2$ (blue curve), the temperature rapidly decreases to $T = 0.5$. When the temperature passes through $T = 0.8$, the particles quickly assemble into a DDQC structure whose $\sigma \approx 0.8$. The test of low initial temperature $T_0 \approx 0.3$ shows that even at such a low temperature, the structure of $\sigma \approx 0.75$ can be formed almost immediately. Then, the policy suggests to keep $T \approx 0.4$ and eventually we get $\sigma > 0.8$. Moreover, the statistical data shown in Fig. S9(d) depicts that the DDQC structure is obtained at $\langle T \rangle = 0.48$ after around 50 steps. Subsequently, both $\langle \sigma \rangle$ and $\langle T \rangle$ maintain their values. Metastable states ($\sigma \in [0.5, 0.7]$) are rarely observed in this isotropic system. This result is in contrast with that of patchy particles shown in the main text. Compared to the DDQC of patchy particles, the RL, in this case, does not feel about the existence of a characteristic temperature T^* although σ increases drastically as $T \approx 0.8$. The agent learns through training that, for isotropic particles, a simpler temperature protocol without a characteristic temperature works for the DDQC formation.

S8 Discussions of reinforcement learning and its hyperparameters

As shown in Fig. 4 in the main text, the convergence to reach the steady state of the iteration for optimising the policy is very slow. Therefore, many epochs are required. Here, we discuss whether we may reduce the computational/training cost using prior knowledge. During training, instead of using random initial temperature T_0 for each epoch, we consider the situation of fixed T_0 either

Parameters	**All T_0 set**	**Low T_0 set**	**High T_0 set**	**Off-policy**
Target, σ^*	0.91	0.91	0.91	0.81
Initial temperature, T_0	[0.2, 1.3]	0.22	1.22	1.22
Number of epochs, N_{epoch}	10, 20, 40, 101	10, 20, 40	10, 20, 40	1
ϵ-greedy	decrease	decrease	decrease	1
Number of steps in each epoch, N_{step}	200	200	200	4000

Table S4: Setting of training sets for DDQC target by patchy particles considering the effect of the number of epochs and prior knowledge of the initial temperature T_0 of each epoch. The value of other hyperparameters in RL is given in Table 1 in the main text.

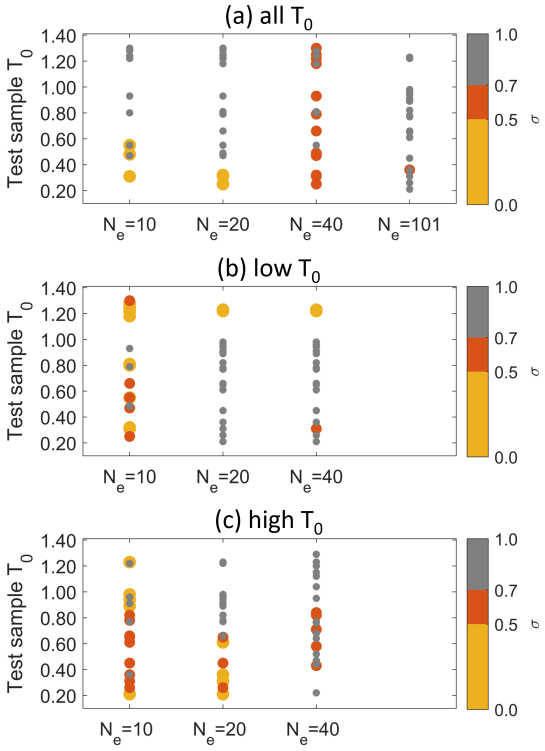

Figure S10: Performance of RL tests at different training sets at the condition (a) all T_0, (b) low T_0 and (c) high T_0 of each epoch in Table S4. The horizontal axis labels indicate the number of epochs N_{e} during training. The vertical axes indicate the random initial temperature T_0 during the tests. The colours of the spots indicate σ values during the tests. The size of the point is for better visualisation. In general, the policy is better (the test has $\sigma > 0.7$) with increasing trained data. Adding constraint on T_0 during training can help to reduce computational cost, however the policy only works well in within the constraint.

at high or low temperature. We also vary the number of epochs for each training set. The policies of those training sets are then evaluated. The conditions and results are summarised in Table S4 and Fig. S10.

In general, the more epochs used during training, the better the policy is; namely, more structures with $\sigma > 0.7$ can be obtained during the test. More importantly, the value of T_0 during training affects the performance of the policy. For example, even when the number of epochs is $N_{\text{e}} = 20$, the policy of 'low T_0' training set works well for the test with low T_0 but fails for the test with high T_0. On the other hand, the policy trained by the 'high T_0' training set works well for the test with high T_0 but fails for the test with low T_0. In other words, if we have a constraint of the initial temperature T_0 in the system, we may reduce the computational cost. This is because the search space during training is smaller. The drawback of the constraint is that the trained policy cannot work outside the constraint.

In this work, we use the Q-table to perform the Q-learning. The DDQC self-assembly has the continuous states σ and T whose essential features can be captured even after discretisation. In this case, the method of Q-table works and is easily implemented. However, the drawback is that the policy is sensitive to such discreteness, as it can not distinguish between the adjacent states and as a result, the agent is trapped in a state-action space and not able to find a better solution. For

example, some testing samples trapped in the metastable states can escape from the metastable states and become DDQC, while others cannot escape. In order to avoid such local traps, one may consider a non-zero epsilon value (finite amplitude of noise) during testing. For example, one may set $\epsilon = 0.2$ for testing so that the action on the temperature is random instead of strictly following the policy. The result in Fig. S11 shows that sometimes adding noise can help the test escape the metastable state and become DDQC. However, $\epsilon = 0.2$ does not always result in DDQC and statistically the test results are not improved by the noise. Therefore, in the current setting, discretisation of the state space does not seem to have serious effects on the estimated policy.

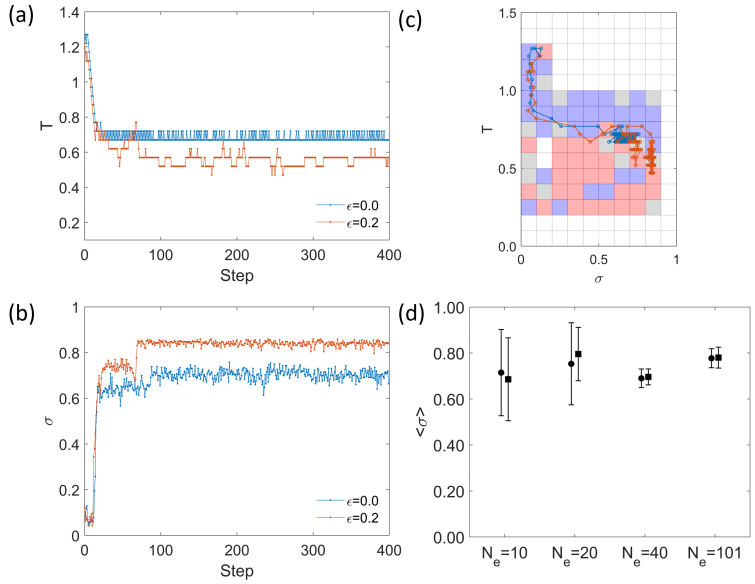

Figure S11: The effect of adding noise on testing. (a,b) Progress of T and σ at $\epsilon = 0$ and $\epsilon = 0.2$ of selected samples for the policy obtained in Fig. 4 and (c) corresponding trajectories in the policy plane. (d) Error bar graph showing the effect of adding noise on the performance of the testing for different training data sets (parameters are given in Table S4 with $\epsilon = 0.0$ (filled circles) and $\epsilon = 0.2$ (filled squares). Sometimes adding noise can help the trajectories escape the metastable state, but the effect is statistically insignificant.

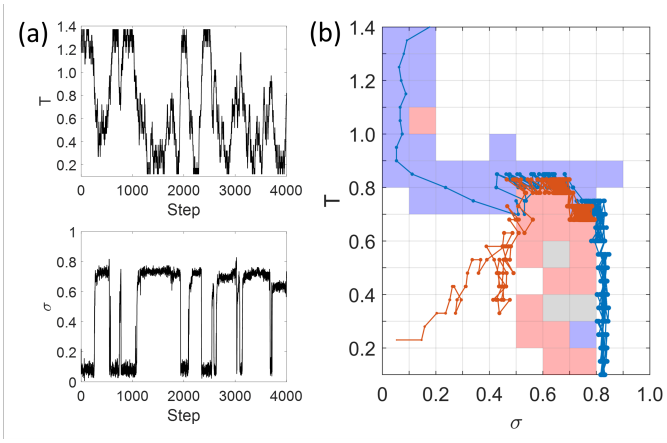

Figure S12: Off-policy training for DDQC in Table S4. (a) The states T and σ during training. (b) Policy and the trajectories of testing samples at different T_0 on the policy plane. The tests start from the left side, and the marker size increases with time. Two regions, decreasing temperature (blue mesh elements) and increasing temperature (red mesh elements), can be seen on the policy plane. The test with high T_0 follows the policy, that is to decrease T to the characteristic temperature T^*, and stay there so that the DDQC structure is formed. The test with low T_0 starts at the inaccessible states. The action applied to the temperature is random until the trajectory hits the accessed states. Then, the trajectory follows the estimated policy and eventually DDQC structure is observed.

So far in this study, the policy is trained with the ϵ-greedy approach. The choice of the action is dependent on the current ϵ and the Q-table. Here we discuss that the RL also works even when the training is off-policy, i.e. during training, the actions at every step are randomly chosen from the action space ($\epsilon = 1$). The result can be found in Fig. S12. Similar to the policy trained with ϵ-greedy in Fig. 4, two regions, decreasing temperature and increasing temperature, divided by the same characteristic temperature can be seen on the policy plane. The tests also show that the DDQC structures are observed. We can see that the number of accessed states in this case is less than the one in Fig. 4. This indicates that the transition probability between the accessed states is higher, and policy at the accessed states is well trained. The non-accessed states have random policies. In our system, these policies work because, after random fluctuations of temperature, the trajectory hits the region of accessed states and then it goes to the characteristic temperature. Eventually, the structure becomes DDQC. For this reason, the completely off-policy training works in our system. Still, we suspect it does not work for other systems.

S9 Stability of the policy

In Section 3.1 we evaluate the policy for dynamic control of the DDQC for the patchy particle system. The policy is trained with 101 epochs. As the training progresses, the policy slowly converges while flipping of policy still occurs, especially in the region of high σ and intermediate T (Fig. 4 in the main text). We test the policy obtained at epoch 81 and find that the tests are statistically similar to those in the policy at epoch 101. As shown in Fig. S13, the temperature quickly reaches the characteristic value, then decreases when $\sigma \approx 0.8$. The finding suggests that though the convergence of the training is not perfect, the estimated policy demonstrates the control strategy that we found as long as the number of epochs is large enough.

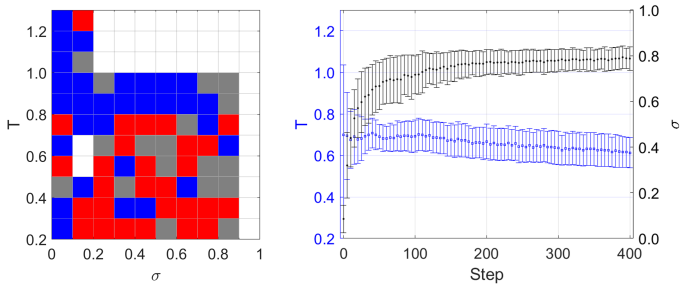

Figure S13: Performance of RL at epoch 81 of the training dataset in Fig. 4. (Left) Policy obtained at epoch 81 and (right) policy evaluation over 40 independent tests.

References

[1] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2 edition, 2022.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 1998.

[3] Sudharsan Ravichandiran. *Deep Reinforcement Learning with Python : Master Classic RL, Deep RL, Distributional RL, Inverse RL, and More with OpenAI Gym and TensorFlow*. Packt Publishing, Limited, Birmingham, 2nd ed. edition, 2020.

[4] Yuji Sugita, Akio Kitao, and Yuko Okamoto. Multidimensional replica-exchange method for free-energy calculations. *The Journal of Chemical Physics*, 113(15):6042–6051, 2000.

[5] YUKITO IBA. Extended ensemble monte carlo. *International Journal of Modern Physics C*, 12(05):623–656, 2001.

[6] Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids*, volume 1. Oxford University Press, 2017.

[7] Uyen Tu Lieu and Natsuhiko Yoshinaga. Formation and fluctuation of two-dimensional dodecagonal quasicrystals. *Soft Matter*, 18(39):7497–7509, 2022.