Estimating Reaction Barriers with Deep Reinforcement Learning

Adittya Pal
Institut for Matematik og Datalogi, Syddansk Universitet, Campusvej 55, 5230 Odense M, Denmark
E-mail: adpal@imada.sdu.dk; ORCID: https://orcid.org/0009-0005-0705-7768

Abstract. Stable states in complex systems correspond to local minima on the associated potential energy surface. Transitions between these local minima govern the dynamics of such systems. Precisely determining the transition pathways in complex and high-dimensional systems is challenging because these transitions are rare events, and isolating the relevant species in experiments is difficult. Most of the time, the system remains near a local minimum, with rare, large fluctuations leading to transitions between minima. The probability of such transitions decreases exponentially with the height of the energy barrier, making the system's dynamics highly sensitive to the calculated energy barriers. This work aims to formulate the problem of finding the minimum energy barrier between two stable states in the system's state space as a cost-minimization problem. It is proposed to solve this problem using reinforcement learning algorithms. The exploratory nature of reinforcement learning agents enables efficient sampling and determination of the minimum energy barrier for transitions.

Keywords: deep reinforcement learning, minimum energy pathways, reaction energy barrier, actor-critic algorithm

1. Introduction

There are multiple sequential decision-making processes one comes across in the world, such as control of robots, autonomous driving, and so on. Instead of constructing an algorithm from the bottom up for an agent to solve these tasks, it would be much easier if one could specify the environment and the state in which the task is considered solved, and let the agent learn a policy that solves the task [24, 31]. Reinforcement learning attempts to address this problem. It is a hands-off approach that provides a feature vector representing the environment and a reward for the actions the agent takes [43]. The objective of the agent is to learn the sequence of steps that maximizes the sum of returns. [42]

One widespread example of a sequential decision-making process where reinforcement learning is utilized is solving mazes [33, 39]. The agent, a maze runner, selects a sequence of actions that might have long-term consequences [49]. Since the consequences of immediate actions might be delayed, the agent must evaluate the actions it chooses and learn to select actions that solve the maze. Particularly, in the case of mazes, it might be relevant to sacrifice immediate rewards for possibly larger rewards in the long term. This is the exploitation-exploration trade-off, where the agent has to learn to choose between leveraging its current knowledge to maximize its current gains or further increasing its knowledge for some possibly larger reward in the long term, possibly at the expense of short-term rewards [7, 40]. The process of learning by an agent while solving a maze is illustrated in Figure 1.

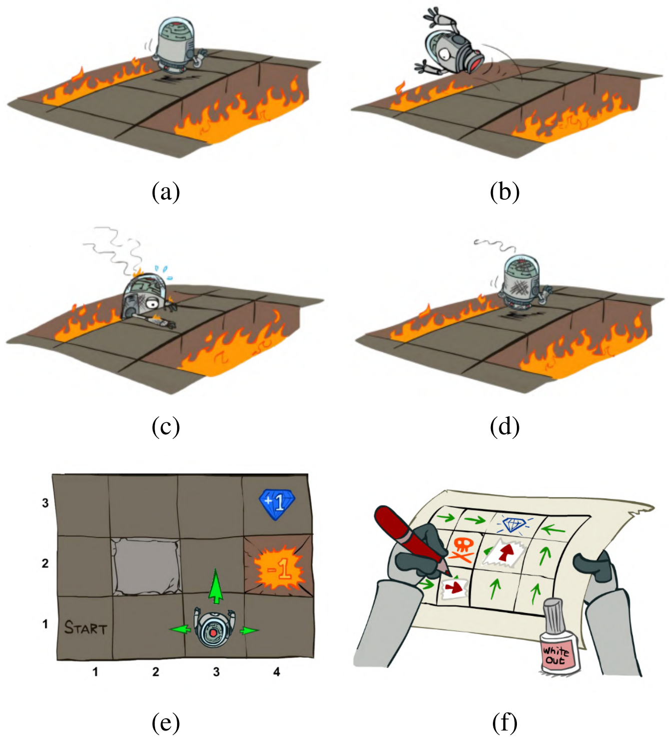


Fig. 1. Maze solving using reinforcement learning: (a) The agent is at a state at a particular time step, and (b) takes an action (according to a policy) to reaches the next state. (c) The agent records the reward obtained by taking the action at that state and (d) continues exploring the environment. After a large number of interactions with the environment (e), the agent learns a policy (f) which maximizes the rewards collected by the agent. The policy (f) gives the sequence of actions that the agent has to take from the initial state to the final state so that it collects the maximum rewards in an episode. The figures are taken from the slides of Dan Klein and Pieter Abbeel for the course CS188 Introduction to Artificial Intelligence at UC Berkeley and available at http://ai.berkeley.edu.

GridWorld is an environment for reinforcement learning that mimics a maze [43]. The agent is placed at the start position in a maze with blocked cells, and the agent tries to reach a stop position with the minimum number of steps possible. One might note an analogy of a maze runner with an agent negotiating the potential energy landscape of a transition event for a system along the saddle point with the minimum height. The start state and the stop state are energy minima on the potential energy surface, separated by an energy barrier for the transition. The agent would have to perform a series of perturbations to the system to take it from one minimum (the start state) to another (the end state) through the located saddle point. In the case of a maze, the (often discrete) action changes the position of the agent (by a fixed measure), but while locating minimum energy pathways, the physical problem demands a continuous action space. However, as in the case of a maze, an action changes the variables describing the system (be it physical coordinates or state variables) by a small measure. As in the maze-solving problem, the agent tries to identify the pathway with the minimum energy barrier. If the number of steps is considered the cost incurred in a normal maze, it is the energy along the pathway that is the cost for the transition event. Reward maps can vary depending on the maze considered, but in the original GridWorld problem, the agent was given a negative reward if the action led to a wall cell, and a zero reward for all non-terminal states (a discount factor < 1 enforces the minimum count of steps). In the case of locating trajectories with a low energy barrier, the agent should be penalized if the action leads to a state with progressively higher energies (but not to an extent that it hinders exploration). The exact reward map used is detailed later in Section 2. A comparison is attempted in Figure 2. A smooth potential energy surface is coarse-grained to construct a maze, where all positions with a negative potential energy are shaded blue (possible move cells), while those with a positive potential energy are shaded red (representing walls). The initial state in the maze is marked yellow, while the final state is marked green. However, instead of classifying a grid cell of the constructed maze either as a wall or a cell, one can discretize the state space and assign an energy value to a cell. An agent can then be trained to reach the final state starting from the initial state and collect the maximum sum of rewards along its path (minimizing the energy along the pathways requires assigning the negative of the energy as the reward for an action leading the agent to the cell). Since this is an episodic problem, one already runs into the problem where the agent moves back and forth between two adjacent cells, collecting rewards from each move in the attempt to maximize the sum of rewards collected, rather than reach the final state and terminate the episode. For this simple setting, the problem is solved by rewarding the agent only the first time it visits a cell and terminating the episode after a fixed number of steps (in this case, 15). The energy profile of the pathway followed by the agent (inferred from the rewards collected in an episode) in this maze is plotted as the dashed green line in Figure 2b. As can be seen, coarse-graining the potential energy surface into an 8×8 maze and then solving it using standard reinforcement learning algorithms provides a reasonable starting point for addressing the problem.

The problem of locating the minimum energy barrier for a transition has applications in physical phase transitions, synthesis plans for materials, activation energies for chemical reactions, and the conformational changes in biomolecules that lead to reactions inside cells. In most of these scenarios, the dynamics are governed by the kinetics of the system (rather than the thermodynamics) because the thermal energy of the system is much smaller than the energy barrier of the transition. This leads to the system spending most of its time around the minima, and some random large fluctuations in the system lead to a transition. This is precisely why transition events are rare and difficult to isolate and characterize with experimental methods. Moreover, these ultra-fast techniques can be applied to only a limited number of systems. Because transition events are rare, sampling them using Monte Carlo methods requires long simulation times, making them inefficient [6]. To sample the regions of the potential energy surface around the saddle point adequately, a large number of samples have to be drawn. Previous work has been done to identify the saddle point and determine the height of the transition barrier—transition path sampling [23], nudged elastic band [19], growing string method [22], to name a few—which use

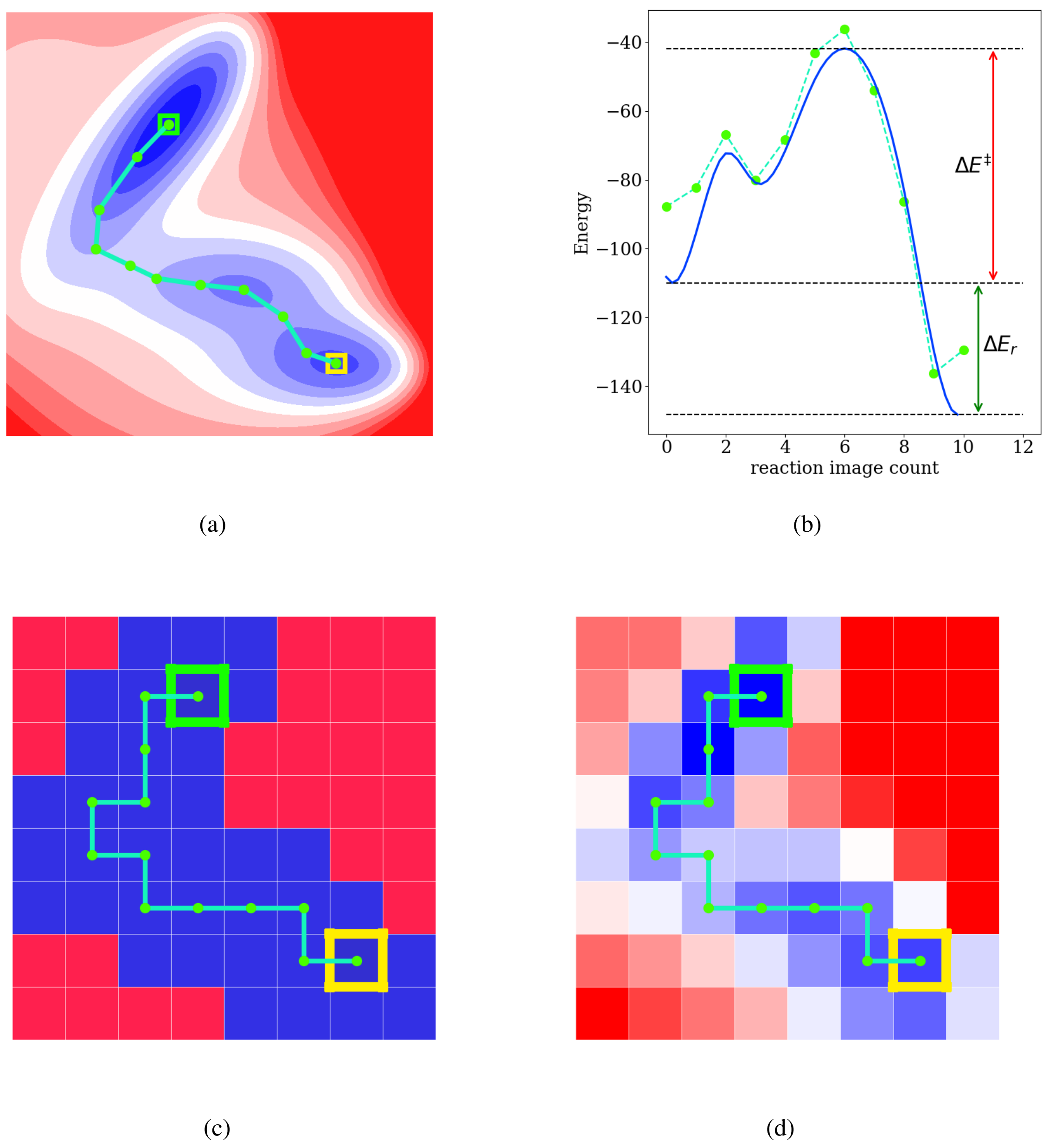


Fig. 2. Estimating reaction barriers by modeling the potential energy surface as a maze: (a) The pathway with the lowest energy barrier as determined by a growing string method on the potential energy surface with 9 intermediate images. (b) The reaction profile, plotted as a solid blue line (interpolated to give a smooth curve) from the pathway determined by the growing string method. The reaction barrier is marked as $\Delta E^{\ddagger}$. Instead of the extreme binary classification of a grid cell as a wall or move as in the maze (c), each cell can be assigned an energy value as in (d).

ideas from gradient descent. However, even for comparatively simple reactions, these methods are not always guaranteed to find the path with the energy barrier that is a global minimum because the initial guess for the pathway might be wrong and lead to a local minimum.

With the advent of deep learning and the use of neural nets as function approximators for complex mappings, there has been increased interest in the use of machine learning [41] to either guess the configuration of the saddle point along the pathway (whose energy can then be determined by standard *ab initio* methods) or directly determine the height of the energy barrier given the two endpoints of the transition. Graph neural networks [47], generative adversarial networks [32], gated recurrent neural networks [8], transformers [30], machine-learned potentials [18, 21], and so on, have been used to optimize the pathway for such transitions.

Noting the superficial similarities between solving a maze and determining the transition pathway with the lowest energy barrier, it is proposed to use standard and tested deep reinforcement learning algorithms used to solve mazes in an attempt to solve the problem of finding minimum energy pathways. The problem is formulated as a min-cost optimization problem in the state space of the system. An actor function approximator suggests the action to be taken by the agent when it is in a particular state. A critic function approximator provides an estimate of the sum of rewards until the end of the episode from the new state after taking the action suggested by the actor. Actor-critic based reinforcement learning techniques have been shown to solve problems effectively, even in higher dimensions [53]. Neural nets are used as the actor and critic function approximators, and a randomly perturbed policy is used to facilitate exploration of the potential energy surface by the agent. Delayed policy updates and target policy averaging are used to stabilize the learning, especially during the first few epochs, which are crucial to the optimal performance of the agent. This formulation is used to determine the barrier height of the optimal pathway in the Müller-Brown potential.

Section 2 describes the methods used to formulate the problem as a Markov decision process and the algorithm used to solve it. Section 3 elaborates on the experiments where the formulated method is used to determine the barrier height of a transition on the Müller-Brown potential. Section 4 contains a short discussion of the work in the context of other similar studies while Section 5 outlines the conclusions drawn from this work.

2. Methods

To solve the problem of finding a pathway with the lowest energy barrier for a transition using reinforcement learning, one has to model it as a Markov decision process. Any Markov decision process consists of (state, action, next state) tuples. In this case, the agent starts at the initial state (a local minimum) and perturbs the system (the action) to reach a new state. Since the initial state was an energy minimum, the current state will have higher energy. However, as in many sequential control problems, the reward is delayed. A series of perturbations that lead to states with higher energies might enable the agent to climb out of the local minimum into another one containing the final state. By defining a suitable reward function and allowing the agent to explore the potential energy surface, it is expected that the agent will learn a path from the initial to the final state that maximizes the rewards. If the reward function is defined properly, it should correspond to the pathway with the lowest energy barrier for the transition.

Once the problem is formulated as a Markov decision process, it can be solved by some reinforcement learning algorithm. In most reinforcement learning algorithms this (state, action, reward, next state, next action) tuple is stored while the agent is learning. Twin Delayed Deep Deterministic Policy Gradient (TD3) [11] is a good start because it prevents the overestimation of the state value function, which often leads to the agent exploiting the errors in the value function and learning a sub-optimal policy. Soft Actor Critic (SAC) [15] tries to blend the deterministic policy gradient with a stochastic policy optimization, promoting exploration by the agent. In practice, using a stochastic policy to tune exploration often accelerates the agent's learning.

2.1. *Markov Decision Process*

The Markov decision process is defined on:

- a state space $\mathcal{S}$, consisting of states $s \in \mathbb{R}^d$, where d is the dimensionality of the system, chosen to be the number of degrees of freedom in the system.
- a continuous action space $\mathcal{A}$, where each action $\Delta s \in \mathbb{R}^d : |(\Delta s)_i| \leqslant 1$ is normalized, and the action is scaled using an appropriate scaling factor λ.

At a state $s^{(k)}$, the agent takes an action $\Delta s^{(k)}$. Since the action is considered a perturbation to the current state of the system, the next state $s^{(k+1)}$ is determined from the current state $s^{(k)}$ as $s^{(k+1)} = s^{(k)} + \lambda \cdot \Delta s^{(k)}$.

To determine the minimum energy barrier for a transition, the reward for an action taking the agent to state $s^{(k+1)}$ from state $s^{(k)}$ is chosen to be the negative of the energy of the next state, $-E\left(s^{(k+1)}\right)$. The negation makes maximizing the sum of rewards collected by the reinforcement learning agent in an episode equivalent to minimizing the sum of energies along the pathway for the transition. The reward acts as immediate feedback to the agent for taking an action in a particular state. However, what is important is the long-term reward, captured by the sum of the rewards over the entire episode, leading the agent to identify a transition pathway with a low sum of energies at all intermediate steps.

Since both the state space and action space are continuous, an actor-critic based method, specifically the soft actor-critic (SAC), is used. Additionally, since the state space is continuous, the episode is deemed to have terminated when the difference between the current state and the target state is smaller than some tolerance, $x \in \mathbb{R}^d : |x - x_t| < \delta$ for some small δ. Otherwise, it would be extremely unlikely that the agent would land exactly at the coordinates of the final state after taking some action. An obvious problem with this definition of the Markov process is that the agent may prefer to remain in a state near the target state (but far enough so that the episode does not terminate), collecting rewards for the rest of the episode. This behavior was observed in Section 3.

2.2. *Algorithm*

SAC, an off-policy learning algorithm with entropy regularization, is used to solve the formulated Markov Decision process because the inherent stochasticity in its policy facilitates exploration by the agent. Entropy regularization tries to balance maximizing the returns till the end of the episode with randomness in the policy driving the agent. The algorithm learns a behavior policy π_θ and two critic Q-functions, which are neural nets with parameters ϕ_1 and ϕ_2 (line 1 of Algorithm 1).

The agent chooses an action $a^{(k)} \equiv \Delta s^{(k)}$ to take when at state $s^{(k)}$ following the policy π_θ (line 8). The returns from the state $s^{(k)}$ when acting according to the policy π is the discounted sum of rewards collected from that step onwards till the end of the episode: $R_t = -\sum_{i=t}^{T} \gamma^{i-t} E(s^{(i)})$. The objective of the reinforcement learning agent is to determine the policy π^* that maximizes the returns, R_t, for states $s \in \mathcal{S}$. This is done by defining a state-action value function, $Q(s^{(i)}, a^{(i)})$, which

Algorithm 1 Computing minimum energy barrier using SAC in environment `env`

```
1: Initialize actor net parameters $\theta$ and critic Q-net parameters $\phi_1$, $\phi_2$
2: Hard update target Q-net parameters: $\phi_{i,\text{target}} \leftarrow \phi_i$ for $i = 1, 2$
3: Initialize replay buffer $\mathcal{R}$
4: training_step = 1
5: for step = 1 to numEpisodes do
6:     state, _ = env.reset()
7:     for t = 0 to maxSteps do
8:         let actor select an action by policy $\pi_\theta$
9:         perturb the action with some noise $\epsilon \sim \mathcal{N}(0, \sigma)$
               $$a^{(k)} = \pi_\theta(s^{(k)}) + \text{clip}(\epsilon, -\epsilon_{\text{lim}}, \epsilon_{\text{lim}})$$
10:        execute action in the env and observe the $\left(s^{(k)}, a^{(k)}, r^{(k)}, s^{(t+1)}, \texttt{done}\right)$ tuple
11:        push it to the replay buffer $\mathcal{R}$
12:        if training_step % agent_update == 0 then
13:            sample a minibatch $\mathcal{B}$ of $(s, a, r, s', \texttt{done})$ tuples from the replay buffer $\mathcal{R}$
14:            compute targets as (where $a' = \pi_\theta(\cdot|s') + \text{clip}(\epsilon, -\epsilon_{\text{lim}}, \epsilon_{\text{lim}})$)
               $$y(r, s') = r + \gamma(1 - \texttt{done})\left(\min_{i=1,2} Q_{\phi_{i,\text{target}}}(s', a') - \alpha \log \pi_\theta(a'|s')\right)$$
15:            update critic Q-nets parameters by one step of gradient descent with loss function (with the gradient clipped by some maximum value)
               $$\frac{1}{|\mathcal{B}|} \nabla_{\phi_i} \left(\sum_{s \in \mathcal{B}} \min_{i=1,2} Q_{\phi_i}(s, a) - y(r, s')\right)^2 \text{ for i = 1, 2}$$
16:            if t % update_target == 0 then
17:                update actor net parameters by one step of gradient descent with loss function (with the gradient clipped by some maximum value)
                   $$\frac{1}{|\mathcal{B}|} \nabla_\theta \left(\sum_{s \in \mathcal{B}} \min_{i=1,2} Q_{\phi_i}(s, \pi_\theta(s)) - \alpha \log \pi_\theta(a|s)\right)^2$$
18:                update the entropy coefficient $\alpha$ as one step of gradient descent with loss function
                   $$\frac{1}{|\mathcal{B}|} \nabla_\alpha \left(\sum_{s \in \mathcal{B}} -\alpha \log \pi_\theta(a|s) - \alpha \log a'\right)^2$$
19:                soft update the target networks: $\phi_{i,\text{target}} \leftarrow \tau\phi_i + (1 - \tau)\phi_{i,\text{target}}$
20:            end if
21:        end if
22:    end for
23: end for
24: return the actor net parameters $\theta$ and critic Q-net parameters $\phi_1, \phi_2$.
```

gives an estimate of the expected returns if action $a^{(i)}$ is taken by the agent when at state $s^{(i)}$: $Q(s^{(i)}, a^{(i)}) = \mathbb{E}\left[R_t : s_t = s^{(i)}, a_t = a^{(i)}\right]$. Since the objective is to maximize the sum of the returns, the action-value function can be recursively defined as

$$Q(s^{(i)}, a^{(i)}) = -E(s^{(i+1)}) + \gamma \max_{a^{(i+1)} \in \mathcal{A}} Q(s^{(i+1)}, a^{(i+1)})$$

which is implemented in line 14 of Algorithm 1.

A replay buffer with a sufficiently large capacity is employed to increase the probability that independent and identically distributed samples are used to update the actor and two critic networks. The replay buffer (in line 3) is modeled as a deque where the first samples to be enqueued (which are the oldest) are also dequeued first, once the replay buffer has reached its capacity and new samples have to be added. Since an off-policy algorithm is used, the critic net parameters are updated by sampling a mini-batch from the replay buffer at each update step (line 13). Stochastic gradient descent is used to train the actor and the two critic nets.

The entropy coefficient α is adjusted over the course of training to encourage the agent to explore more when required and exploit its knowledge at other times (line 18) [16]. However, some elements from the TD3 algorithm [11] are borrowed to improve the learning of the agent, namely delayed policy updates and target policy smoothing. Due to the delayed policy updates, the critic Q-nets are updated more frequently than the actor and the target Q-nets to allow the critic to learn faster and provide more precise estimates of the returns from the current state. To address the problem of instability in the learning, especially in the first few episodes while training the agent, target critic nets are used. Initially, the critic nets are duplicated (line 2) and subsequently soft updates of these target nets are carried out after an interval of a certain number of steps (line 19). This provides more precise estimates for the state-action value function while computing the returns for a particular state in line 14. Adding noise to the inputs during the training of a machine learning model often leads to improved performance because it acts as an L_2-regularizer [5] and prevents the model from memorizing patterns in the training data sample in supervised learning scenarios. Similarly, TD3 adds noise to the action predicted by the actor network to smooth out the Q-function, so that the agent does not memorize the imprecise estimates of the Q-function early on during the training process. This prevents the policy from exploiting the imprecise estimates of the Q-function approximator for certain actions, reducing the chances of learning a brittle policy that does not generalize well. The logic is that for well-behaved, smooth reward maps, the reward should not abruptly change with small differences in the action. The addition of clipped noise to the action chosen by the actor net (in line 9) also encourages the agent to explore the potential energy surface. The changes to the SAC algorithm, borrowed from TD3, are highlighted in blue in the pseudocode of Algorithm 1. The parameters used in the particular implementation of the algorithm are listed in Table 1.

3. Experiments

The proposed algorithm is applied to determine the pathway with the minimum energy barrier on the Müller–Brown potential energy surface [37]. The Müller–Brown potential has been used to benchmark the performance of several algorithms that determine the minimum energy pathways, such as the molecular growing string method [12], Gaussian process regression for nudged elastic bands [26], and

	Parameter	Value
$Q_\phi(s,a)$	network architecture	4-256-256-1
	activation for hidden layer	*relu*
	activation for output layer	none
	learning rate	10^{-4}
$\pi_\theta(s)$	network architecture	2-256-256-2
	activation for hidden layer	*relu*
	activation for output layer	none
	learning rate	10^{-4}
Agent	τ or Polyak averaging parameter	0.005
	γ or discount factor	$1 - 10^{-2}$
	λ or scaling factor for actions	0.01
	optimizer	Adam
	replay buffer $\mathcal{R}$ capacity	10^4
	minibatch size for update	128
	maximum steps per episode	500
	number of training epochs	10000
SAC specific	initial α or entropy coefficient	0.5
	α value	variable
	learning rate for α	10^{-4}
TD3 specific	target update delay interval	8 steps
	actor noise standard deviation	0.4
	actor noise clip	1.0

Table 1
Parameters used while training the RL agent.

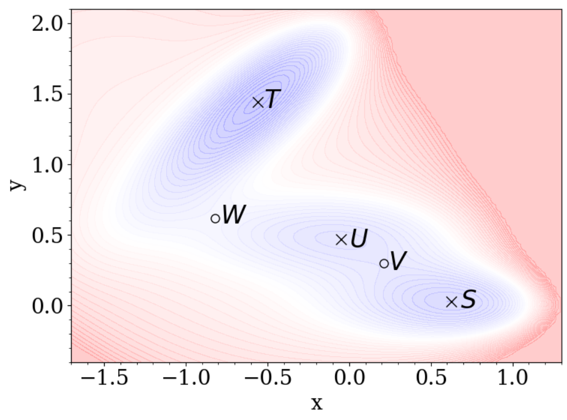
accelerated molecular dynamics [46]. Therefore, it is also used in this work to demonstrate the applicability of the proposed method. A custom `Gym` environment [44] was created following the gymnasium interface (inheriting from the `class Gym`) to model the problem as a Markov Decision Process to be solved by a reinforcement learning pipeline. The values for the parameters used in Algorithm 1 are listed in Table 1.

3.1. Results

The Müller–Brown potential is characterized by the following potential:

$$V(x,y) = \sum_{i=0}^{3} W_i \cdot \exp\left[a_i\,(x-\overline{x}_i)^2 + b_i\,(x-\overline{x}_i)\,(y-\overline{y}_i) + c_i\,(y-\overline{y}_i)^2\right] \tag{1}$$

where $W = (-200,-100,-170,15)$, $a = (-1,-1,-6.5,0.7)$, $b = (0,0,11,0.6)$, $c = (-10,-10,-6.5,0.7)$, $\overline{x} = (1,0,-0.5,-1)$, and $\overline{y} = (0,0.5,1.5,1)$. The potential energy surface for the system is plotted in Figure 3a, and the coordinates of the local minima and saddle points for the potential energy surface and their corresponding energies are tabulated in Table 3b. The RL agent was trained to locate a path on this surface from S $(0.623, 0.028)$ with an initial random step (with zero mean and a standard deviation of 0.1) taken as the starting state to T $(-0.558, 1.442)$ as the terminal state, with the minimum energy barrier. The first random step was chosen to avoid the same starting point in



(a) Potential energy surface to find the paths with minimum energy barrier.

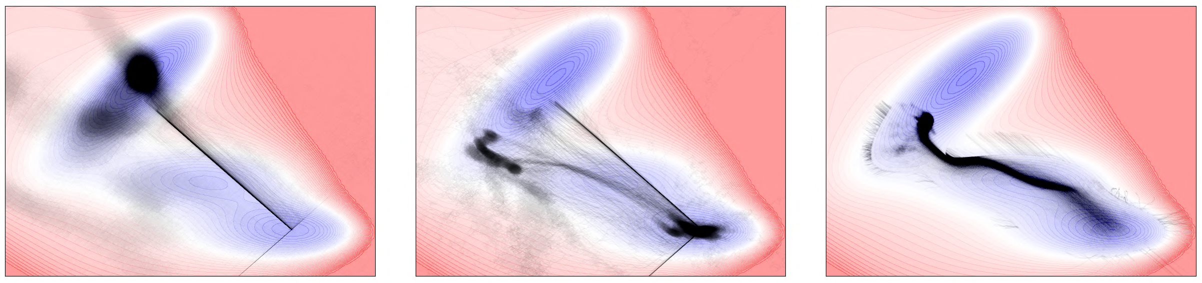
	pt	x	y	E
Minima	S	0.623	0.028	-108.2
	T	-0.558	1.442	-146.7
	U	-0.050	0.467	-80.8
Saddle points	V	0.210	0.300	-72.3
	W	-0.820	0.620	-40.7

(b) Minima and saddle points on the chosen potential energy surface.

Parameter	Value
number of dimensions (d)	2
limits for dimensions 1	$(-1.70, 1.30)$
limits for dimensions 2	$(-0.40, 2.10)$
scaling factor for action (λ)	0.01
tolerance for convergence (δ)	10^{-4}

(c) Some parameters of the Markov Decision process to find pathways with the minimum energy barrier on the chosen potential.

Fig. 3. The environment in which the agent learns to find the path with the minimum energy barrier.

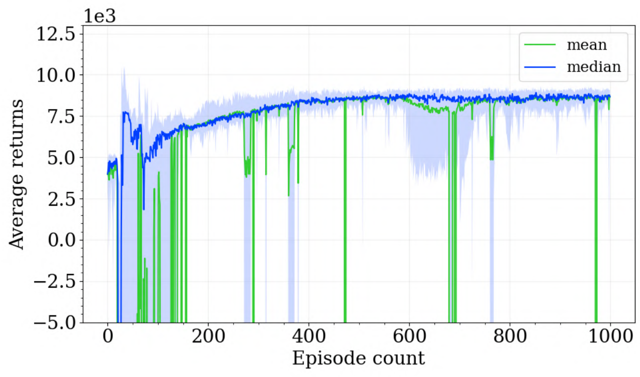


(a) TD3 (b) SAC (c) Algorithm 1

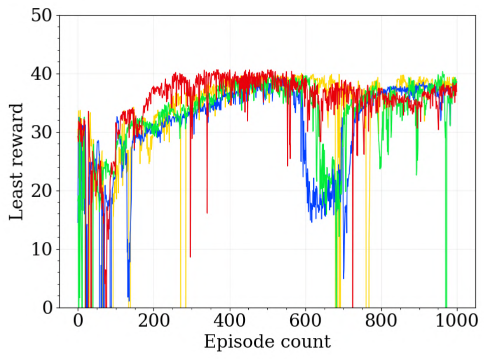
Fig. 4. Scatter plot of the regions visited by the reinforcement learning agent during the course of learning while using different algorithms.

each training iteration of the agent, so it learns a more generalized policy. Some of the parameters for the Markov Decision Process to model this potential are given in Table 3c.

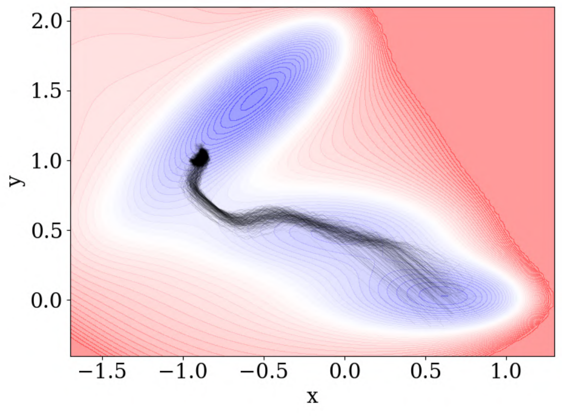
Figure 4 shows scatter plots of the trajectories generated by various reinforcement learning algorithms: TD3 in Figure 4a, SAC in Figure 4b, and the proposed modified SAC algorithm in Figure 4c. While the agent trained by the TD3 algorithm does reach the intended target state, it starts exploiting a flaw in the formulation of the MDP by trying to reach the vicinity of the final state quickly and staying near enough to it so that it collects rewards but does not terminate the episode. This results in a high density in the plot along the straight line connecting the initial and final states and around the final state. It gives a much higher estimate than the correct minimum energy barrier for the transition. The agent trained using SAC shows improved performance, possibly due to the entropy regularizer forcing it to learn a more diverse policy (rather than one that would result in a straight line connecting the initial and final states). However, while generating trajectories in the testing environment, most of the trajectories did not leave the local minima in the vicinity of the start state. Moreover, the learned policy has high variance.



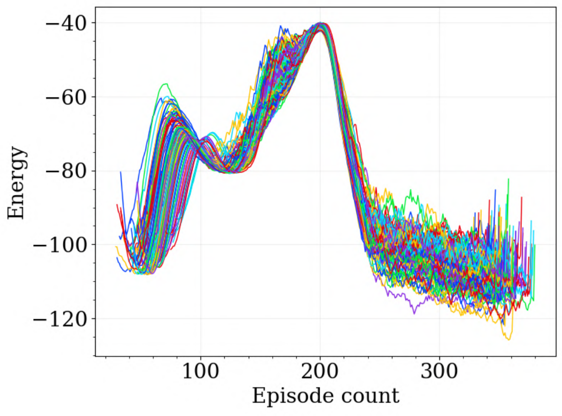
(a) Learning curve for the agent averages over 11 trials.



(b) Least reward collected in the episode.



(c) Trajectories generated by the agent after training.



(d) Energy profiles for the generated trajectories.

Fig. 5. (a) The learning curve for the agent in the reinforcement learning environment. (b) The plot of the variation of the least reward collect by the agent in a step with the validation episode count. (c) Trajectories generated by the trained agent following the learnt policy along with the corresponding energy profiles (d).

The proposed Algorithm 1 learns a much more stable policy and confines itself to exploring the region with lower energies leading to the terminal state (specifically the vicinity of the saddle point) rather than the entire environment. It explores sufficiently and then exploits the state-action values learned appropriately, providing better estimates of the energy barrier for the transition.

The learning curve for the agent under Algorithm 1 is shown in Figure 5a. The data for this curve was generated by allowing the agent to solve the MDP in evaluation mode once every 10 training episodes, where the neural networks were not updated to monitor the agent's learning. The ascending learning curve indicates that the agent gradually learns to find a path to the terminal state that maximizes the rewards. The blue line represents the median reward, while the green line shows the mean reward over 11 training iterations, each consisting of 10×1000 training episodes. The light blue shaded region denotes the spread of the rewards (maximum and minimum). A low spread in the rewards indicates consistent performance by the reinforcement learning agent in the validation episodes.

In Figure 5c, an ensemble of paths generated by the trained RL agent with the starting points slightly perturbed from $(0.623, 0.028)$ by noise added from $\mathcal{N}(0, 0.1)$ is plotted on the potential energy surface. The model used to generate these trajectories was the model at the 500th validation step (and not after the entire training consisting of 1000 validation steps) for reasons elaborated later. It can be seen from Figure 5c that the agent spends more time near the terminal state rather than reaching the terminal state to get an immediate reward, as it allows the agent to collect rewards for more steps. In the case of a coarse-grained maze representation of the potential energy surface, this problem was solved by rewarding the agent only the first time it visited a grid cell. Using a similar idea of not rewarding the agent when it is in the close neighborhood of an already visited state artificially perturbs the reward map and did not work in this case. The best performance was achieved by gradually varying the maximum number of steps the agent was allowed to take in an episode. If the number of steps allowed in an episode is too short, the agent does not escape the local minima to reach the terminal state. If the number of steps is too large, then the agent reaches the terminal state and discovers that it receives larger rewards by remaining in its vicinity, but not so close that the episode is terminated. In an attempt to reach the terminal state earlier, the agent tries to approach the terminal state sooner, choosing a more direct pathway, which lifts the trajectory out of the saddle point. It was observed that a maximum of 500 steps per episode led to the best performance by the agent. The agent did not leave the local minima if fewer steps were allowed (200 steps), and the agent passed through states with much higher energies than optimal to reach a minimum energy state if longer episodes were allowed (1000 steps).

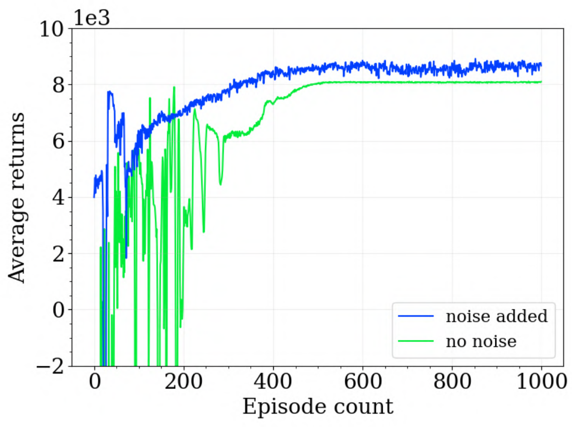
Early stopping during the training of a neural network has often been found to be helpful in scenarios where continued training worsens the model's performance [1, 38]. Borrowing the idea of early stopping, the agent's training was stopped when the minimum reward collected by the agent (corresponding to the maximum energy along the pathway) started increasing again. Such behavior was observed in the case of the agent, as plotted in Figure 5b. The minimum reward collected by the agent during the episode increases initially (indicating that the agent finds a pathway with a progressively better energy barrier) until the 500th validation step before decreasing slightly. Plots for only 4 of the 11 trials are shown for clarity. This might indicate that the agent does not improve its performance after that step. Additionally, the learning curve in Figure 5a shows an increase in the spread after 500 iterations. These reasons led to using the model after 500 iterations for generating the final trajectory in test mode to estimate the energy barrier for the reaction. The energy profiles along the generated trajectories are plotted in Figure 5d aligned by the maximum of the profiles (and not by the start of the trajectories) for better visualization. The predicted energy barrier for the transition of interest is -40.36 ± 0.21. One can see that the agent learns to predict the path with the correct minimum energy barrier, albeit the energy barrier estimated by the agent is a little higher than the optimal analytical solution (-40.665). However, the result demonstrates that reinforcement learning algorithms can be used to locate the minimum energy barrier for transitions between stable states in complex systems. The paths suggested by the trained agent cluster around the minimum energy path and pass through the vicinity of the actual saddle point representing the energy barrier. However, there still seems to be some way to go to improve the sampling densities around the saddle point, which determines the barrier height, to avoid overestimating it.

3.2. Ablation Studies

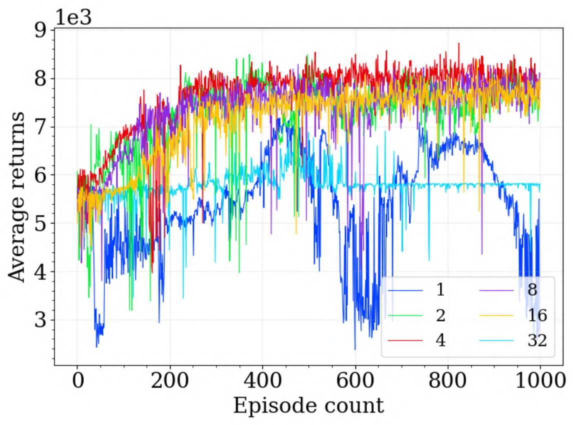
Several modifications were made to the standard SAC algorithm to be used in this particular case (highlighted in blue in Algorithm 1). Studies were performed to understand the contribution of each individual component to the working of the algorithm in this particular environment by comparing the performance of the algorithm with different hyperparameters for a component. The parameters for one

Modification	Value	Returns
target policy smoothing	absent	8082 ± 10
	present	**8651 ± 107**
policy update delay	0	4826 ± 984
	2	7738 ± 234
	4	7741 ± 220
	8	**7994 ± 206**
	16	7677 ± 204
	32	5783 ± 142
α tuning	10^{-3}	3877 ± 29
	10^{-2}	4153 ± 14
	10^{-1}	7425 ± 414
	2×10^{-1}	4880 ± 654
	5×10^{-1}	6247 ± 190
	tunable	**8651 ± 107**

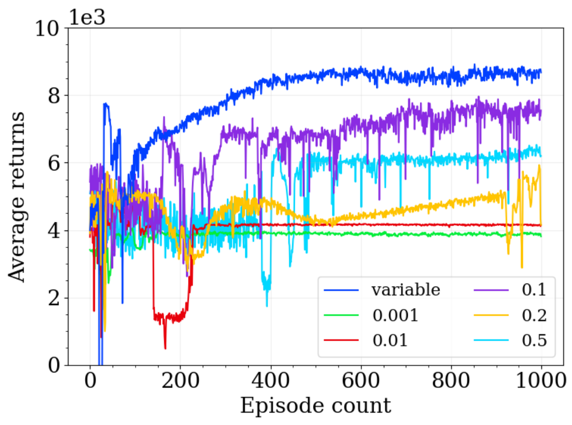
(a) Returns for the last 100 episodes of trials for each modification on the learning of the agent.



(b) Effect of adding noise to the target action on the learning of the agent.



(c) Effect of delaying the update of the actor net, and target critic-nets on the learning of the agent.



(d) Effect of varying constant values of α and a tunable α on the learning of the agent.

Fig. 6. Effect of the various modifications to the SAC algorithm on the learning of the agent.

modification was varied, keeping the parameters for the other two modifications unchanged from the fine-tuned algorithm. Each modification and its contribution to the overall learning of the agent are described in the following sections. The mean and the standard deviation of the returns from the last 100 training steps for each modification to the existing algorithm are listed in Table 6a to compare the performance of the agents. The modification that leads to the highest returns is highlighted.

3.2.1. Target Policy Smoothing

Injecting random noise (with a standard deviation σ) into the action used in the environment (in line 9 of Algorithm 1) encourages the agent to explore, while adding noise to the actions used to calculate the targets (in line 14 of Algorithm 1) acts as a regularizer, forcing the agent to generalize over similar actions. In the early stages of training, the critic Q-nets can assign inaccurate values to some state-action pairs, and the addition of noise prevents the actor from rote learning these actions based on incorrect feedback. On the other hand, to avoid the actor taking a too random action, the action is clipped by some maximum value for the noise (as done in lines 9 and 14 of Algorithm 1). The effect of adding noise to spread the state-action value over a range of actions is plotted in Figure 6b. Adding noise leads to the agent learning a policy with less variance in the early learning stages and a more consistent performance.

3.2.2. Delayed Policy Updates

Delaying the updates for the actor nets and the target Q-nets (in lines 17 and 19 of Algorithm 1) allows the critic Q-nets to update more frequently and learn at a faster rate, so that they can provide a reasonable estimate of the value for a state-action pair before it is used to guide the policy learned by the actor net. The parameters of the critic Q-nets might often change abruptly early on while learning, undoing whatever the agent had learned (catastrophic failure). Therefore, delayed updates of the actor net allow it to use more stable state-action values from the critic nets to guide the policy learned by it. The effect of varying intervals of delay for the actor update on the learning of the agent is plotted in Figure 6c. Updating the actor net for every update of the critic nets led to a policy with a high variance (blue plot). Delaying the update of the actor net to once every 2 updates of the critic resulted in the agent learning a policy that provided higher returns but still had a high variance (green plot). Delaying the update of the actor further (once every 4 and 8 updates of the critic net plotted as the red and magenta curves, respectively) further improved the performance of the agent. One can notice the lower variance in the policy of the agent during the early stages (first 200 episodes of the magenta curve) for the agent which updates the actor net and target critic nets once every 8 updates of the critic nets. However, delaying the updates for too long intervals would cripple the learning of the actor. The performance of the agent suffers when the update of the actor is delayed to once every 16 updates of the critic nets (yellow curve) and the agent fails to learn when the update of the actor net is further delayed to once every 32 updates of the critic nets (cyan curve).

3.2.3. Tuning the Entropy Coefficient

The entropy coefficient α can be tuned as the agent learns (as done in line 18 of Algorithm 1), which overcomes the problem of finding the optimal value for the hyperparameter α [16]. Moreover, simply fixing α to a single value might lead to a poor solution because the agent learns a policy over time: it should still explore regions where it has not learned the optimal action, but the policy should not change much in regions already explored by the agent that have higher returns. In Figure 6d, the effect of the variation of the hyperparameter α on the learning of the agent is compared. As can be seen, a tunable α allows the agent to learn steadily, encouraging it to explore more in the earlier episodes and exploiting the returns from these explored regions in the latter episodes, resulting in a more stable learning curve (blue curve). A too low value of α, such as 10^{-3} or 10^{-2}, makes the algorithm more deterministic (TD3-like), which leads to sub-optimal performance and the agent being stuck in a local minimum (plotted as green and red curves, respectively). An α value of 0.1 has comparable performance to the tunable α, but the learning curve is less stable and there are abrupt changes in the policy function (magenta curve). The original implementation of SAC suggested 0.2 as a fixed value for α, which leads to a learning curve resulting in a policy with high variance (yellow curve). A too high value of α, such as 0.5, makes the algorithm more stochastic (REINFORCE-like), which also leads to sub-optimal learning (cyan curve).

4. Discussions

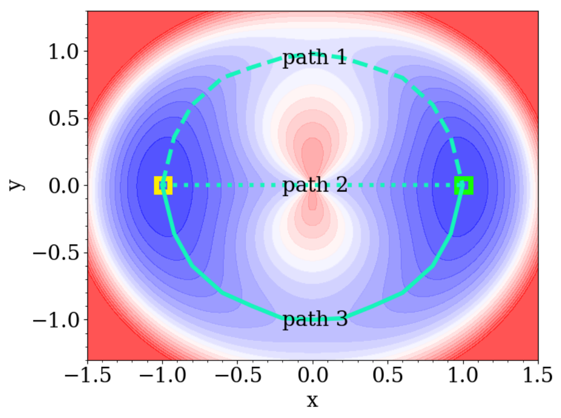
Previous works in determining transition pathways using deep learning or reinforcement learning techniques include formulating the problem as a shooting game solved using deep reinforcement learning

[52]. The authors in [52] sample from higher energy configurations and shoot trajectories with randomized initial momenta in opposite directions, expecting them to converge at the two desired local minima. In contrast, the method proposed here starts from a minimum on the potential energy surface and attempts to generate a trajectory to another minimum. Additionally, in [20], the problem is formulated as a stochastic optimal control problem, where neural network policies learn a controlled and optimized stochastic process to sample the transition pathway using machine learning techniques. Stochastic diffusion models have also been used to model elementary reactions and generate the structure of the transition state, preserving the required physical symmetries in the process [9]. Furthermore, the problem of finding transition pathways was recast into a finite-time horizon control optimization problem using the variational principle and solved using reinforcement learning in [14]. Moreover, a hybrid-DDPG algorithm was implemented in [35] to identify the global minimum on the Müller–Brown potential, but did not identify pathways between minima as done in this work. Recent work [2] used an actor-critic reinforcement learning framework to optimize molecular structures and calculate minimum energy pathways for two reactions.

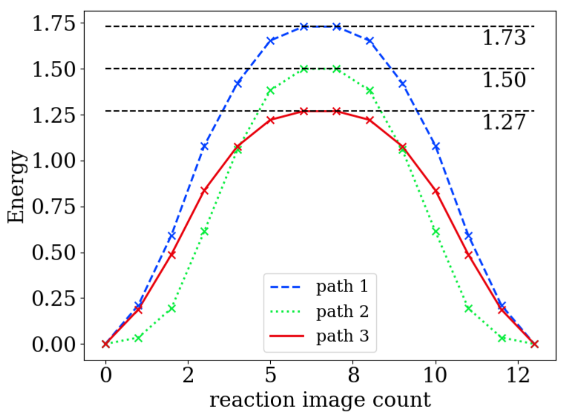
There has also been previous work [54] to optimize chemical reactions by perturbing the experimental conditions to achieve better selectivity, purity, or cost for the reaction using deep reinforcement learning. While this approach has macroscopic applications in laboratory settings, the method proposed here focuses on a much narrower problem: given a potential energy surface, how well can the minimum energy barrier be estimated for a transition between two minima? Deep reinforcement learning has also been used to find a minimum energy pathway consisting of multiple elementary transitions in catalytic reaction networks [27]. While the free energy barrier for a transition (which is mapped to a reward) between two states is calculated using density functional theory (DFT) with VASP software in [27, 28], the objective of the proposed method in this work is to estimate that free energy barrier using an agent trained via deep reinforcement learning, requiring no quantum mechanical calculations. Additionally, reinforcement learning techniques are implemented in [51] to minimize the cost of synthesis pathways (consisting of multiple elementary transitions) by considering the price of the starting molecules and the atom economy of individual transitions. Furthermore, a reinforcement learning approach is used to search for process routes that optimize economic profit for a Markov decision process modeling the thermodynamic state space as a graph.

5. Conclusion

Advancements in reinforcement learning algorithms based on the state-action value function have led to their application in diverse sequential control tasks such as Atari games, autonomous driving, robot movement control and more physical domains [3, 4, 13, 48]. This project formulated the problem of finding the minimum energy barrier for a transition between two local minima as a cost minimization problem, solved using a reinforcement learning setup with neural networks as function approximators for the actor and critics. A stochastic policy was employed to facilitate exploration by the agent, further perturbed by random noise. Target networks, delayed updates of the actor, and a replay buffer were used to stabilize the learning process for the reinforcement learning agent. While the proposed framework samples the region around the saddle point sufficiently, providing a good estimate of the energy barrier for the transition, there is definitely scope for improvement. The method has been applied only to a two-dimensional system, but as future work, it could be extended to more realistic and higher-dimensional systems. One promising alternative would be to use max-reward reinforcement learning [45], as it aligns



(a) A potential energy surface with pathways passing through three different saddle points.



(b) The energy profiles of the three pathways depicted in (a) with their respective barrier heights.

Fig. 7. (a) Three possible pathways on a potential energy surface passing though different saddle points and (b) the energy profile for the three possible transition pathways.

well with the objective of maximizing the minimum reward obtained in an episode. However, a drawback of this method is that the reinforcement learning agent must be trained from scratch if one needs to find minimum energy pathways on a different potential energy surface. In other words, an agent trained on one potential energy surface cannot be used to determine the minimum energy barrier on a different surface, similar to how an agent trained in one `Gymnasium` environment cannot solve tasks in another. Another limitation is that the agent can only locate pathways to minima lower than the starting minimum; otherwise, remaining in the starting minimum would yield higher rewards for the agent.

This work differs from previous works using reinforcement learning [2, 14, 20, 52] by providing a much simpler formulation of the problem, using the energy of the state directly as the reward while searching for transition pathways with the minimum energy barrier. One of the main advantages of using a reinforcement learning based method is that, unlike traditional methods such as the nudged elastic band or the growing string method, it does not require an initial guess for the trajectory. Traditional methods use energy gradient information along the trajectory to iteratively improve to a trajectory with better energetics. However, the success of these methods depends on the initial guess for the trajectory, and gradient based methods might get stuck in a local minimum. As shown in the constructed potential energy surface taken from [10] in Figure 7, there may be multiple saddle points between two minima. The trajectory to which a nudged elastic band or growing string method converges depends on the initial guess for the starting trajectory. Typically, the initial guess trajectory is a simple linear interpolation between the starting and ending points, which leads to the dotted trajectory (Path 2 with a barrier of 1.50 units). Traditional gradient-based methods report this trajectory as the optimal one because the local gradients along the trajectory are minimal and cannot be improved by perturbation. However, the reinforcement learning-based method proposed in this work identifies the trajectory represented by a solid line (Path 3 with a barrier of 1.27 units) as the minimum energy pathway. The suboptimal solution overestimates the energy barrier for the transition by $(150-127)/127$ or 18%, and hence underestimates the frequency with which it occurs by $1 - e^{-1.50-(-1.27)} = 20\%$. Underestimates of the probability for a transition to occur would leading to imperfect modeling of the dynamics of the system. The use of a stochastic policy in a reinforcement learning setup avoids this problem, increasing the chances of finding a better estimate of the transition barrier as the agent explores the state space. However, as a trade-off for the simple model and generic approach, the agent learns slowly, requires a large number of environment interactions, and would have to retrained to work in a new potential energy surface.

Acknowledgements

The author would like to gratefully acknowledge the computational resources provided by Institut for Matematik og Datalogi, Syddansk Universitet for this work. The author would also like to thank the two reviewers for their insightful suggestions to improve the manuscript.

References

[1] Y. Bai, E. Yang, B. Han, Y. Yang, J. Li, Y. Mao, G. Niu and T. Liu, Understanding and Improving Early Stopping for Learning with Noisy Labels, in: *Advances in Neural Information Processing Systems*, Vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang and J.W. Vaughan, eds, Curran Associates, Inc., 2021, pp. 24392–24403. https://proceedings.neurips.cc/paper_files/paper/2021/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf.

[2] R. Barrett and J. Westermayr, Reinforcement Learning for Traversing Chemical Structure Space: Optimizing Transition States and Minimum Energy Paths of Molecules, *The Journal of Physical Chemistry Letters* **15**(1) (2024), 349–356, PMID: 38170921. doi:10.1021/acs.jpclett.3c02771.

[3] C. Beeler, U. Yahorau, R. Coles, K. Mills, S. Whitelam and I. Tamblyn, Optimizing thermodynamic trajectories using evolutionary and gradient-based reinforcement learning, *Phys. Rev. E* **104** (2021), 064128. doi:10.1103/PhysRevE.104.064128.

[4] C. Beeler, S.G. Subramanian, K. Sprague, C. Bellinger, M. Crowley and I. Tamblyn, Demonstrating ChemGymRL: An Interactive Framework for Reinforcement Learning for Digital Chemistry, in: *AI for Accelerated Materials Design - NeurIPS 2023 Workshop*, 2023. https://openreview.net/forum?id=cSz69rFRvS.

[5] C.M. Bishop, Training with Noise is Equivalent to Tikhonov Regularization, *Neural Computation* **7**(1) (1995), 108–116. doi:10.1162/neco.1995.7.1.108.

[6] P.G. Bolhuis and D.W.H. Swenson, Transition Path Sampling as Markov Chain Monte Carlo of Trajectories: Recent Algorithms, Software, Applications, and Future Outlook, *Advanced Theory and Simulations* **4**(4) (2021), 2000237. doi:https://doi.org/10.1002/adts.202000237. https://onlinelibrary.wiley.com/doi/abs/10.1002/adts.202000237.

[7] G. Brunner, O. Richter, Y. Wang and R. Wattenhofer, Teaching a Machine to Read Maps With Deep Reinforcement Learning, *Proceedings of the AAAI Conference on Artificial Intelligence* **32**(1) (2018). doi:10.1609/aaai.v32i1.11645. https://ojs.aaai.org/index.php/AAAI/article/view/11645.

[8] S. Choi, Prediction of transition state structures of gas-phase chemical reactions via machine learning., *Nat Commun* **14** (2023). doi:10.1038/s41467-023-36823-3.

[9] C. Duan, Y. Du, H. Jia and H.J. Kulik, Accurate transition state generation with an object-aware equivariant elementary reaction diffusion model, *Nature Computational Science* **3**(12) (2023), 1045–1055. doi:10.1038/s43588-023-00563-7.

[10] W. E, W. Ren and E. Vanden-Eijnden, Simplified and improved string method for computing the minimum energy paths in barrier-crossing events, *The Journal of Chemical Physics* **126**(16) (2007), 164103. doi:10.1063/1.2720838.

[11] S. Fujimoto, H. van Hoof and D. Meger, Addressing Function Approximation Error in Actor-Critic Methods, 2018. https://arxiv.org/abs/1802.09477.

[12] A. Goodrow, A.T. Bell and M. Head-Gordon, Transition state-finding strategies for use with the growing string method, *The Journal of Chemical Physics* **130**(24) (2009), 244108. doi:10.1063/1.3156312.

[13] S. Gow, M. Niranjan, S. Kanza and J.G. Frey, A review of reinforcement learning in chemistry, *Digital Discovery* **1** (2022), 551–567. doi:10.1039/D2DD00047D.

[14] J. Guo, T. Gao, P. Zhang, J. Han and J. Duan, Deep reinforcement learning in finite-horizon to explore the most probable transition pathway, *Physica D: Nonlinear Phenomena* **458** (2024), 133955. doi:https://doi.org/10.1016/j.physd.2023.133955. https://www.sciencedirect.com/science/article/pii/S0167278923003093.

[15] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018. https://arxiv.org/abs/1801.01290.

[16] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine, Soft Actor-Critic Algorithms and Applications, 2019. https://arxiv.org/abs/1812.05905.

[17] H. Hanke and D. Knees, A phase-field damage model based on evolving microstructure, *Asymptotic Analysis* **101** (2017), 149–180.

[18] S. Heinen, G.F. von Rudorff and O.A. von Lilienfeld, Transition state search and geometry relaxation throughout chemical compound space with quantum machine learning, *The Journal of Chemical Physics* **157**(22) (2022), 221102. doi:10.1063/5.0112856.

[19] G. Henkelman, B.P. Uberuaga and H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, *The Journal of Chemical Physics* **113**(22) (2000), 9901–9904. doi:10.1063/1.1329672.

[20] L. Holdijk, Y. Du, F. Hooft, P. Jaini, B. Ensing and M. Welling, Stochastic Optimal Control for Collective Variable Free Sampling of Molecular Transition Paths, 2023. https://arxiv.org/abs/2207.02149.

[21] R. Jackson, W. Zhang and J. Pearson, TSNet: predicting transition state structures with tensor field networks and transfer learning, *Chem. Sci.* **12** (2021), 10022–10040. doi:10.1039/D1SC01206A.

[22] M. Jafari and P.M. Zimmerman, Reliable and efficient reaction path and transition state finding for surface reactions with the growing string method, *Journal of Computational Chemistry* **38**(10) (2017), 645–658. doi:https://doi.org/10.1002/jcc.24720. https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.24720.

[23] H. Jung, R. Covino, A. Arjun et al., Machine-guided path sampling to discover mechanisms of molecular self-organization., *Nat Comput Sci* **3** (2023), 334–345–. doi:10.1038/s43588-023-00428-z.

[24] L.P. Kaelbling, M.L. Littman and A.W. Moore, Reinforcement learning: a survey, *J. Artif. Int. Res.* **4**(1) (1996), 237–285–.

[25] A. Khan and A. Lapkin, Searching for optimal process routes: A reinforcement learning approach, *Computers & Chemical Engineering* **141** (2020), 107027. doi:https://doi.org/10.1016/j.compchemeng.2020.107027. https://www.sciencedirect.com/science/article/pii/S0098135420303999.

[26] O.-P. Koistinen, F.B. Dagbjartsdóttir, V. Ásgeirsson, A. Vehtari and H. Jónsson, Nudged elastic band calculations accelerated with Gaussian process regression, *The Journal of Chemical Physics* **147**(15) (2017), 152720. doi:10.1063/1.4986787.

[27] T. Lan and Q. An, Discovering Catalytic Reaction Networks Using Deep Reinforcement Learning from First-Principles, *Journal of the American Chemical Society* **143**(40) (2021), 16804–16812, PMID: 34606265. doi:10.1021/jacs.1c08794.

[28] T. Lan, H. Wang and Q. An, Enabling high throughput deep reinforcement learning with first principles to investigate catalytic reaction mechanisms., *Nat Commun* **15**(6281) (2024). doi:https://doi.org/10.1038/s41467-024-50531-6.

[29] E. Lefever, A hybrid approach to domain-independent taxonomy learning, *Applied Ontology* **11**(3) (2016), 255–278.

[30] K.-D. Luong and A. Singh, Application of Transformers in Cheminformatics, *Journal of Chemical Information and Modeling* **64**(11) (2024), 4392–4409, PMID: 38815246. doi:10.1021/acs.jcim.3c02070.

[31] P. Maes, Modeling Adaptive Autonomous Agents, *Artificial Life* $\mathbf{1}(1_2)(1993), 135 - -162. doi : 10.1162/artl.1993.1.1\_2.135.$

[32] M.Z. Makoś, N. Verma, E.C. Larson, M. Freindorf and E. Kraka, Generative adversarial networks for transition state geometry prediction, *The Journal of Chemical Physics* **155**(2) (2021), 024116. doi:10.1063/5.0055094.

[33] T. Mannucci and E.-J. van Kampen, A hierarchical maze navigation algorithm with Reinforcement Learning and mapping, in: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8. doi:10.1109/SSCI.2016.7849365.

[34] P.S. Meltzer, A. Kallioniemi and J.M. Trent, Chromosome alterations in human solid tumors, in: *The Genetic Basis of Human Cancer*, B. Vogelstein and K.W. Kinzler, eds, McGraw-Hill, New York, 2002, pp. 93–113.

[35] A.W. Mills, J.J. Goings, D. Beck, C. Yang and X. Li, Exploring Potential Energy Surfaces Using Reinforcement Machine Learning, *Journal of Chemical Information and Modeling* **62**(13) (2022), 3169–3179, PMID: 35709515. doi:10.1021/acs.jcim.2c00373.

[36] P.R. Murray, K.S. Rosenthal, G.S. Kobayashi and M.A. Pfaller, *Medical Microbiology*, 4th edn, Mosby, St. Louis, 2002.

[37] K. Müller and L.D. Brown, Location of saddle points and minimum energy paths by a constrained simplex optimization procedure., *Theoret. Chim. Acta* **53** (1979), 75–93. doi:10.1007/BF00547608.

[38] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak and I. Sutskever, Deep Double Descent: Where Bigger Models and More Data Hurt, in: *International Conference on Learning Representations*, 2020. https://openreview.net/forum?id=B1g5sA4twr.

[39] D. Osmanković and S. Konjicija, Implementation of Q — Learning algorithm for solving maze problem, in: *2011 Proceedings of the 34th International Convention MIPRO*, 2011, pp. 1619–1622.

[40] E. Parisotto and R. Salakhutdinov, Neural Map: Structured Memory for Deep Reinforcement Learning, 2017. https://arxiv.org/abs/1702.08360.

[41] G.M. Rotskoff, A.R. Mitchell and E. Vanden-Eijnden, Active Importance Sampling for Variational Objectives Dominated by Rare Events: Consequences for Optimization and Generalization, in: *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, J. Bruna, J. Hesthaven and L. Zdeborova, eds, Proceedings of Machine Learning Research, Vol. 145, PMLR, 2022, pp. 757–780. https://proceedings.mlr.press/v145/rotskoff22a.html.

[42] D. Silver, S. Singh, D. Precup and R.S. Sutton, Reward is enough, *Artificial Intelligence* **299** (2021), 103535. doi:https://doi.org/10.1016/j.artint.2021.103535. https://www.sciencedirect.com/science/article/pii/S0004370221000862.

[43] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, A Bradford book, MIT Press, 1998. ISBN 9780262193986. https://books.google.dk/books?id=CAFR6IBF4xYC.

[44] M. Towers, J.K. Terry, A. Kwiatkowski, J.U. Balis, G.d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J.J. Tai, A.T.J. Shen and O.G. Younis, Gymnasium, Zenodo, 2023. doi:10.5281/zenodo.8127026. https://zenodo.org/record/8127025.

[45] G. Veviurko, W. Böhmer and M. de Weerdt, To the Max: Reinventing Reward in Reinforcement Learning, 2024. https://arxiv.org/abs/2402.01361.

[46] P.R. Vlachas, J. Zavadlav, M. Praprotnik and P. Koumoutsakos, Accelerated Simulations of Molecular Systems through Learning of Effective Dynamics, *Journal of Chemical Theory and Computation* **18**(1) (2022), 538–549, PMID: 34890204. doi:10.1021/acs.jctc.1c00809.

[47] B. Wander, M. Shuaibi, J.R. Kitchin, Z.W. Ulissi and C.L. Zitnick, CatTSunami: Accelerating Transition State Energy Calculations with Pre-trained Graph Neural Networks, 2024. https://arxiv.org/abs/2405.02078.

[48] M. Wen, E.W.C. Spotte-Smith, S.M. Blau et al., Chemical reaction networks and opportunities for machine learning., *Nat Comput Sci* **3** (2023), 12–24–. doi:10.1038/s43588-022-00369-z.

[49] M.A. Wiering and H. van Hasselt, Ensemble Algorithms in Reinforcement Learning, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **38**(4) (2008), 930–936. doi:10.1109/TSMCB.2008.920231.

[50] E. Wilson, Active vibration analysis of thin-walled beams, PhD thesis, University of Virginia, 1991.

[51] C. Zhang and A.A. Lapkin, Reinforcement learning optimization of reaction routes on the basis of large, hybrid organic chemistry–synthetic biological, reaction network data, *React. Chem. Eng.* **8** (2023), 2491–2504. doi:10.1039/D2RE00406B.

[52] J. Zhang, Y.-K. Lei, Z. Zhang, X. Han, M. Li, L. Yang, Y.I. Yang and Y.Q. Gao, Deep reinforcement learning of transition states, *Phys. Chem. Chem. Phys.* **23** (2021), 6888–6895. doi:10.1039/D0CP06184K.

[53] X. Zhang, Actor-Critic Algorithm for High-dimensional Partial Differential Equations, 2020. https://arxiv.org/abs/2010.03647.

[54] Z. Zhou, X. Li and R.N. Zare, Optimizing Chemical Reactions with Deep Reinforcement Learning, *ACS Central Science* **3**(12) (2017), 1337–1344, PMID: 29296675. doi:10.1021/acscentsci.7b00492.

The text added in the re-submission is in red. The count of works in the .tex file was ~ 9000, which was below the target of 12000 words for a submission.

I thank reviewer 1 for the positive comments. The reply to the reviewer's suggestions are listed below.

(1) The computational methods should be described in greater technical detail. Of the parameters in Tables 1 and 2, only the choice of the entropy coefficient α is covered in depth. As a minimum the choice of values for the Polyak averaging parameter τ, discount factor γ and scaling factor λ should be explained. The reasons for the choice of neural network architecture could also be discussed.

The choice of the values for the parameters τ and γ, and the neural network architecture were kept the same as the TD3 and SAC algorithm implementations. The scaling factor λ adjusts the step size for the agent, and it was this combination of the scaling factor and number of steps in an episode which led to the best performance of the agent. The agent reaches near the terminal state with just the number of required small enough steps to end the episode. A larger value of λ resulted in the agent taking longer steps over regions of the potential energy surface with a higher energy to give an incorrect estimate of the barrier height. Smaller values of λ led to smaller steps, and the agent did not leave the local minima to explore other regions of the potential energy surface, and is unsuccessful a its assigned task. It is difficult to visualizing these effects as a plot of the rewards against the number of validation steps as those in Figure 6.

(2) The captions of the figures contain a lot of useful information which may be better presented in the body of the text. Examples include the sentence "In most reinforcement learning algorithms this (state, action, reward, next state, next action) tuple is stored while the agent is learning" from Figure 1, the final sentence under Figure 3, and all of the discussion on Figure 4 which is not referenced at all in the main text. The final sentence of the caption for Figure 2 could also be moved, but should also be rewritten for clarity and contains two typos ("course" and "ans").

The captions of all the figures, and Figure 1, 3 and 4 in particular have been shortened. The typos in the caption of Figure 2 has been corrected.

(3) Page 5 paragraph 2 describes how the actor and critic functions are to be approximated without having introduced these terms first.

A sentence introducing the actor and critic functions has been added (Page 5, lines 14-18).

(4) Figure 7 may benefit from clarifying which pathway each energy profile refers to, as it was not immediately obvious to me that the plotting characters alone (dotted/dashed/solid lines) identify this.

The three pathways are labeled and the labels are used to identify the energy profiles of the respective pathways.

I thank reviewer 2 for the constructive comments and suggestions. The reply to the reviewer's suggestions are listed below.

(1) A solution is only presented for a (likely unrealistically) simple version of the problem here. While a new RL algorithm is presented, there are only limited discussions comparing it with the state-of-the-art existing methods. The paper itself feels rushed and unrefined.

While I admit that the Müller–Brown potential is a constructed artificial potential, nevertheless it had been used to show the effectiveness of the algorithms is use to determine minimum energy pathways:

- *Growing String Methods*: Wolfgang Quapp; A growing string method for the reaction pathway defined by a Newton trajectory. J. Chem. Phys. 1 May 2005; 122 (17): 174106. https://doi.org/10.1063/1.1885467
- *Nudged Elastic Band*: Graeme Henkelman, Hannes Jónsson; Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. J. Chem. Phys. 8 December 2000; 113 (22): 9978–9985. https://doi.org/10.1063/1.1323224 used a two dimensional LEPS model potential with two minima only (Müller–Brown potential has an intermediate third minima).
- Baron Peters, Andreas Heyden et. al, A growing string method for determining transition states: Comparison to the nudged elastic band and string methods. J. Chem. Phys. 1 May 2004; 120 (17): 7877–7886. https://doi.org/10.1063/1.1691018
- *Accelerated Molecular Dynamics*: Adaptively Accelerating Reactive Molecular Dynamics Using Boxed Molecular Dynamics in Energy Space, Robin J. Shannon, Silvia Amabilino et. al, Journal of Chemical Theory and Computation 2018 14 (9), 4541-4552. DOI: 10.1021/acs.jctc.8b00515.
- *Artificial Force Induced Reaction*: Quapp W, Bofill JM, Mechanochemistry on the Müller–Brown surface by Newton trajectories. Int J Quantum Chem. 2018;118:e25522. 10.1002/qua.25522
- *Reinforcement Learning*: Exploring Potential Energy Surfaces Using Reinforcement Machine Learning, Alexis W. Mills, Joshua J. Goings et. al, Journal of Chemical Information and Modeling 2022 62 (13), 3169-3179, DOI: 10.1021/acs.jcim.2c00373 uses a RL agent to explore the potential energy surface.

The discussions section has been rewritten to compare it with the state-of-the-art existing methods.

(2) Below are extra comments/suggestions for improving the paper. Introduction: Could use more citations when discussing what RL is. (Pg. 2 Ln. 3-18)

Some more citations have been added while introduction reinforcement learning in Section 1.

(3) Figure 1 doesn't really add any value or insight (especially given how much space it takes up and that it is not an original figure to this manuscript). The introduction would benefit from just including the figure caption in the main text instead.

The figure has been downsized to occupy less space and its source is acknowledged.

(4) Making the comparison of navigating a maze to navigating a potential energy landscape is useful for making the problem accessible to readers in either field but I feel it could be taken further. Currently the author connects start/end states in these two scenarios but the analogy could benefit from connecting other components as well (such as actions and rewards) to complete the analogy. (Pg. 2 Ln. 19-29)

While it was a little difficult to make an analogy between the actions and rewards (especially rewards because the agent has different objectives in both environments), it has been added at lines 5-8 and 11-16 on page 3.

(5) Typo in Figure 2 caption “ans solving then solving it using standard reinforcement”. (Pg. 4 Ln. 46)

The typo was removed and most of the caption is incorporated into the main text.

(6) Methods: There seems to be a typo in the equation for Rt (extra comma). (Pg. 6 Ln. 30)

The typo was corrected.

(7) State-action function is missing a gamma. (Pg. 6 Ln. 38)

The missing γ was added.

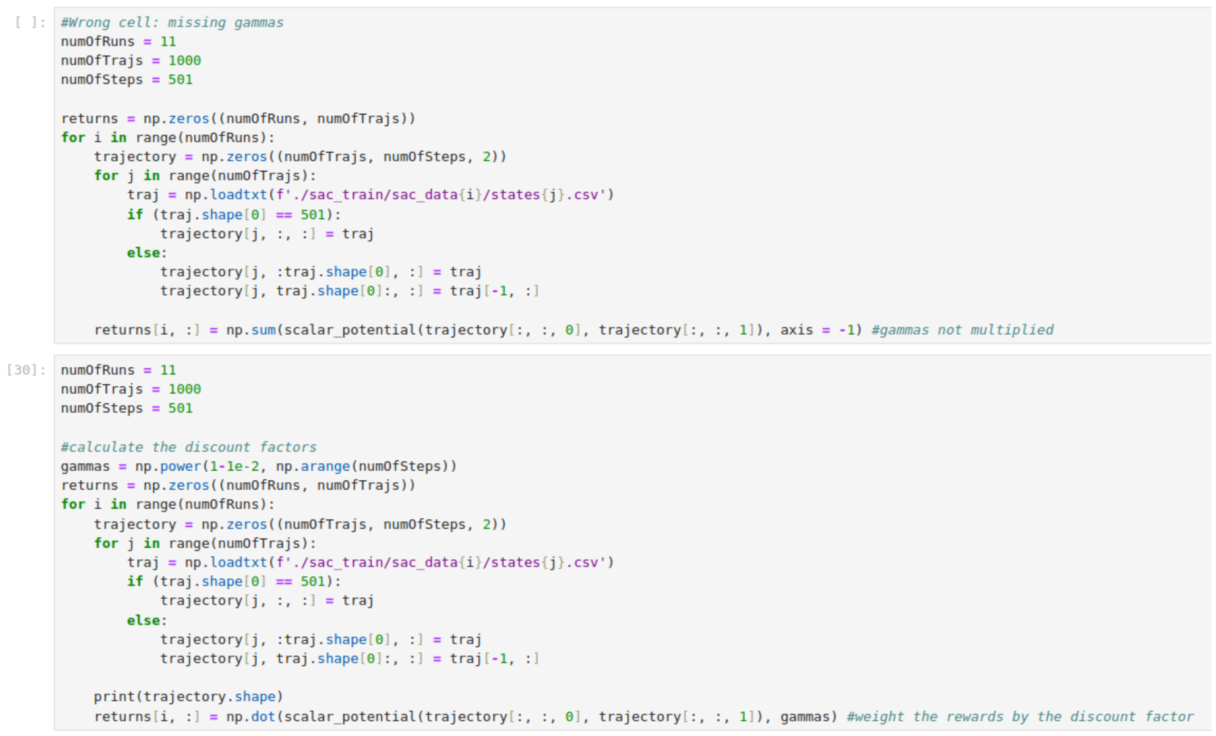
(8) In the methods section (2), there is a mix of presenting the basic ideas of RL in detail (which assumes the readers are not familiar with RL) and glossing over more advanced concepts such as actor-critic methods and target policy smoothing (which assumes the readers are extremely familiar with RL). The author defines these concepts much later but it should be done in this section (rather than in Experiments).

A sentence introducing the actor and critic functions has been added on page 5, lines 14-18. Target policy smoothing is introduced on page 8 lines 25-33.

(9) There are some clear issues with how the MDP is defined and what the author’s intentions of an optimal solution are. The episode terminates when the current state and target state are equal (within some tolerance δ) giving the agent an immediate (and final) reward of 146.7. The reward function encourages the agent to minimize the sum of energies over the states it occupies at each time step. The agent could sit in some state δ away from the target state and receive rewards of $146.7 - \epsilon$ for infinite steps (the results shown in Figure 5a support this). While the point of this study is to determine reaction barriers (and not necessarily the entire optimal reaction pathway), the reward function isn’t necessarily encouraging the agent to find the minimum energy barrier. With this reward function, one could imagine a chemical landscape where it is more beneficial for the agent to pass through states with much higher energies than the optimal reaction barrier because it allows the agent to reach the minimum energy states much faster. Thus this set-up does not guarantee one would find the optimal reaction barrier energy nor the optimal reaction pathway.

The author acknowledges that formulated MDP suffers from the problem raised by the reviewer. The number of steps in an episode, the scaling factor of actions and the number of training epochs were varied to come up with a set of values for these three parameters which minimize the problems cause due to the imperfect formulation of the MDP. The aim is to discourage the agent from sitting in some state δ away from the target state, and collect rewards for the remainder of the episode by truncating the episode after 500 steps. A small scaling factor λ was used for the actions, so that the agent does not make too many long jumps through higher energy states to reach a state with lower energy faster. Decreasing λ would require increasing the maximum number of steps in an episode, so that the agent explores regions away from the starting point, but not too much that the trajectory passes through regions with higher energy. The lowest reward in the episode (corresponding to the highest energy along the pathway, plotted in Figure 5b) was monitored to decide when the agent stops improving at its intended task. Using the model after 1000 validation steps indeed led to a higher estimate of the energy barrier.

(10) Experiments: Figure 3b is not referenced in the text and seems unrelated to Figure 3a. It would be much better paired with the results shown in Figure 5. This figure claims that the agent is achieving an average return of 55,000, however based on how the author defined the return in this environment, this is impossible. Even if the agent started in the state with a global minimum energy of -146.7 (which it does not), using the discount factor provided in Table 1 (1-10-2), the maximum possible theoretical return would be: $\sum 146.7 * 0.99^n <= 14670$. This does not take into account that the episode ends if the agent reaches this state (which would lower the theoretical maximum even further).



```
[ ]: #Wrong cell: missing gammas
     numOfRuns = 11
     numOfTrajs = 1000
     numOfSteps = 501

     returns = np.zeros((numOfRuns, numOfTrajs))
     for i in range(numOfRuns):
         trajectory = np.zeros((numOfTrajs, numOfSteps, 2))
         for j in range(numOfTrajs):
             traj = np.loadtxt(f'./sac_train/sac_data{i}/states{j}.csv')
             if (traj.shape[0] == 501):
                 trajectory[j, :, :] = traj
             else:
                 trajectory[j, :traj.shape[0], :] = traj
                 trajectory[j, traj.shape[0]:, :] = traj[-1, :]

         returns[i, :] = np.sum(scalar_potential(trajectory[:, :, 0], trajectory[:, :, 1]), axis = -1) #gammas not multiplied
```

```
[30]: numOfRuns = 11
      numOfTrajs = 1000
      numOfSteps = 501

      #calculate the discount factors
      gammas = np.power(1-1e-2, np.arange(numOfSteps))
      returns = np.zeros((numOfRuns, numOfTrajs))
      for i in range(numOfRuns):
          trajectory = np.zeros((numOfTrajs, numOfSteps, 2))
          for j in range(numOfTrajs):
              traj = np.loadtxt(f'./sac_train/sac_data{i}/states{j}.csv')
              if (traj.shape[0] == 501):
                  trajectory[j, :, :] = traj
              else:
                  trajectory[j, :traj.shape[0], :] = traj
                  trajectory[j, traj.shape[0]:, :] = traj[-1, :]

          print(trajectory.shape)
          returns[i, :] = np.dot(scalar_potential(trajectory[:, :, 0], trajectory[:, :, 1]), gammas) #weight the rewards by the discount factor
```

The figure was shifted to be a part of Figure 5, as suggested by the reviewer. I would like to acknowledge my mistake of not multiplying the discount factor γ while calculating the returns from the episodes which has been corrected. It leads to a much flatter learning curve.

(11) Figure 4 is not referenced in the text at all. This figure suggests the author did more experiments comparing TD3, SAC, and their hybrid algorithm of the two, however there are no other indications of this work.

Some text is added on Pages 10 and 11 to elaborate on the plots in Figure 4.

(12) I appreciate the experiments shown in Figure 6. They provide some justifications to the modifications made to SAC but are not sufficient for the introduction of a new RL algorithm.

The table in figure 6 was updated after multiplying the discount factor while calculating the average rewards.

(13) Figure 7 is only offhandedly mentioned in the Conclusions, despite being arguably the most interesting result of the paper.

Text was added to page 16, lines 38 onward, to elaborate on the results from Figure 7. However, it was kept a part of conclusions because it demonstrates an advantage of a reinforcement learning based approach compared to the existing gradient based algorithms.

(14) This section should be split. There should be a discussion on the results presented separate from the conclusions of the paper.

The section has been split. Discussions contain comparison of the current work with previous work, while conclusions focus only on this work.

(15) There are several very relevant studies in the intersection of chemical reactions and RL not cited in this work that need to be addressed. At the very least, a discussion needs to be added that compares these works with the work presented here and explains the relative novelty of it.

I would like to thank out the reviewer for pointing out relevant references. A discussion of these in the context of the current work is added in Section 4.

(16) There are only two examples of potential energy surfaces shown in this paper and both are 2-dimensional. While these are convenient for visualization purposes, I would imagine one of the major advantages of this approach is that it could be applied to higher dimensional systems with little modification.

I acknowledge that two-dimensional (simpler) environments have been as examples. Higher dimensional state spaces would require more computational resources and longer training-times for the agent to learn. I would like to point out that most state-of-the art works also use two-dimensional models. One of the references suggested bu the reviewer (which was already present in the first submission), Zhang, Jun, et al. Deep reinforcement learning of transition states." Physical Chemistry Chemical Physics 23.11 (2021): 6888-6895, uses *four two dimensional* models, all with two potential wells. While all systems, except the first, have multiple dimensions, two dimensions have been chosen (by expert knowledge, called order parameters), and the agent used only those two dimensions. To avoid this (human) choice, environments with (only) two dimensions were used. When higher dimensions are used in Lan, Tian, and Qi An. "Discovering catalytic reaction networks using deep reinforcement learning from first-principles." Journal of the American Chemical Society 143.40 (2021): 16804-16812, the state space is discrete.

(17) The paper feels rushed and disjointed. There's not a clear flow of the study as there are methods explained in Experiments and results shown Discussions and Conclusions. While the problem and algorithm are worth presenting, there is not enough results shown for either.

Several sections of the paper have been rewritten for clarity.