Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Learning Shortcuts in the Chemical Space: A Search for High Performance Single Molecule Magnet Candidates via Reinforcement Learning

James O'Reilly

supervised by Dr. Alessandro Lunghi

B.A. Theoretical Physics
Final Year Capstone Project
School of Physics
Trinity College Dublin
March 25, 2024

Abstract

The discovery of single molecule magnets in 1993 offered the possibility to store information on the molecular scale, changing the perspective of how data can be stored. In particular, single molecule magnets store information as a bit by retaining their magnetisation, however they have only been shown to function at extremely low temperatures, hindering any practical implications. Recent research has been focused on discovering high performance single molecule magnets that operate at higher temperatures. This project attempts to demonstrate that a reinforcement learning approach has the potential to be a highly efficient method of exploring the chemical space for high performance single molecule magnets. The Q-learning algorithm was implemented to optimize datasets of molecules for the most promising high performance single molecule magnet candidates. This approach was shown to be effective for synthetically generated datasets, with significant improvements for larger datasets. However, the algorithm failed to be effective for small and incomplete physical datasets. Evidence has been shown that under appropriate conditions, the Q-learning algorithm has the potential to be a powerful tool in exploring the chemical space for novel high performance single molecule magnets at low computational cost, and warrants further research.

Acknowledgements

This research project would not have been possible without the exceptional support and insights of my supervisor, Alessandro Lunghi and his research group. In particular, I wish to extend a massive thank you to Lion, Zahra and Lorenzo. You always made time for me and your help and encouragement was immensely appreciated.

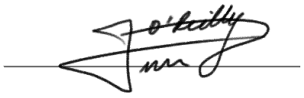
To my friends and family, you should know that your support and belief in me was worth so much more than I can express on paper.

This work was supported by RIT (Research IT, Trinity College Dublin). The majority of calculations were performed on the Boyle cluster maintained by the Trinity Centre for High Performance Computing. This cluster was funded through grants from the European Research Council and Science Foundation Ireland.

Plagiarism Declaration

I have read and understood the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also read and understood the guide, and completed the 'Ready Steady Write' Tutorial on avoiding plagiarism, located at https://libguides.tcd.ie/academic-integrity/ready-steady-write.

Signed: 

Contents

1 Introduction **1**

2 Background & Theory **2**
2.1 Single Molecule Magnets . . . 2
2.1.1 Physics of SMMs . . . 2
2.2 Reinforcement Learning . . . 4
2.2.1 Markov Decision Processes . . . 5
2.2.2 Q-Learning . . . 9
2.3 Computational Method . . . 10

3 Results & Discussion **12**
3.1 Simple Synthetic Data . . . 12
3.1.1 Q-learning Improvement . . . 16
3.2 Complex Synthetic Data . . . 18
3.3 Physical Data . . . 20

4 Conclusion **23**

Appendix **24**
A.1 Figures . . . 24
A.2 Source Code . . . 24
A.3 Derivations . . . 25

Bibliography **25**

1

Introduction

We currently live in a time where nearly all information is electronically stored. It is easy to forget that only 50 years ago, floppy disks were the gold standard of data storage. Despite their large dimensions, around 8 inches, they barely had enough capacity to store a simple pdf a few megabytes in size. Technology has evolved rapidly over the last few decades, where the standard of storage density now lies at $\sim$ 200GB in^{-2}[1] in hard drives. The change has been immense and the progress of humankind as we know it would not have been possible without these innovations.

Single molecule magnets offer the potential to store information at the molecular scale. Implementation of single molecule magnets would revolutionize the field of data storage, and would be hailed as a major breakthrough. Unfortunately, since their initial discovery in 1993, single molecule magnets have only been shown to reliably store information at extremely low temperatures and widespread implementation remains unfeasible. Research has been ongoing in an attempt to discover high performance single molecule magnets that perform effectively at higher temperatures, with the ultimate goal of achieving ambient room temperature performance.

This project aims to contribute to this search by demonstrating the potential of a reinforcement learning approach in rapidly exploring the chemical space to find promising high temperature single molecule magnet candidates at low computational cost. The operational temperature has a high dependence on an intrinsic property of the molecule called axial magnetic anisotropy, which is extremely challenging and expensive to compute. It is therefore not at all a straightforward task to analyse a large number of molecules for their potential to be effective single molecule magnets. The use of reinforcement learning is hoped to significantly reduce the number of these expensive computations required to find high performance single molecule magnets.

In particular, a well-known reinforcement learning algorithm known as Q-learning was implemented to construct molecules step by step in an attempt to learn what creates the best performing single molecule magnets. The degree to which a molecule was considered to be promising or not was determined by analysis of this magnetic anisotropy property. In this way, investigative efforts can be focused towards the most promising high temperature single molecule magnet candidates, while analysis of a large number of inferior molecules can be avoided, drastically reducing the computational demand of exploring the chemical space.

2

Background & Theory

2.1 Single Molecule Magnets

Single molecule magnets (SMMs) are molecules that display a magnetic memory effect. In fact, they are the smallest material unit with memory effect, making it possible to store information at the molecular level. This greatly exceeds the performance of the current technology standard of storing information in hard disk drives (HDDs). HDDs use thin magnetic layers consisting of a large number of magnetic domains that are individually magnetised up or down to store information in binary as a bit. The size of these magnetic domains are typically on the order of a few hundreds of nanometers, whereas SMMs are only tens of angstroms in length, yielding a factor of 100 improvement in storage density. It should also be mentioned that there is a limit to the size of the magnetic domains in HDDs, they cannot be made infinitely smaller. This is known as the superparamagnetic limit and is due to the magnetic domains becoming unstable and susceptible to spontaneous flips in their orientation due to thermal fluctuations at small sizes, resulting in information loss. In contrast, SMMs do not suffer from this drawback, however they have a serious limitation of their own that must be overcome. Currently SMMs are only effective in retaining their magnetisation at extremely low temperatures, with the best performing SMMs achieving an operational temperature of just above liquid nitrogen temperature ($\sim 80K$). For SMMs to have any practical implications, one with close to room temperature performance must be discovered.

2.1.1 Physics of SMMs

Single molecule magnets display superparamagnetic behaviour at the molecular scale. In particular, SMMs possess an axial magnetic anisotropy property as a result of intrinsic factors. This implies a connection between the magnetic properties and the structure of the molecule. Consequently, there exists an energetically favourable direction of magnetisation, known as the easy axis.

Upon the application of an external magnetic field, the SMM becomes magnetised along the easy axis. If the field is then removed and the temperature of the system remains below a certain blocking temperature T_B, the magnetization is retained in the same direction. It would then require a strong external magnetic field in the opposite direction to flip the magnetisation of the SMM to be directed antiparallel along the easy axis. This describes how SMMs exhibit a magnetic hysteresis loop (see Figure 2.1).

It can be seen that in the 0 applied field configuration there are two possible states, with magnetisation parallel and antiparallel to the easy axis respectively. In other words, there exists a spin up or spin down state that is stable and information can be stored.

However, if the blocking temperature T_B is exceeded, the system has enough thermal energy to overcome an energy barrier U and relaxation of the magnetisation back to equilibrium occurs, resulting in the loss of the stored information. This energy barrier depends on the anisotropy parameter D given by $U = S^2|D|$, where S is the ground state spin value of the molecule. The ground state is multi-fold degenerate with $2S+1$ degeneracy. However, due to zero field splitting (ZFS) the $2S+1$ sublevels are separated by energy that follows the expression $E(M_S) = M_S^2 \cdot D$, where $M_S = 0, \pm1, \pm2, \ldots, \pm S$ is the magnetic spin quantum number. From this expression, it can be seen that for $D < 0$ the low-lying energy levels are the ones with highest $|M_S|$ values. In particular, this results in a bistable ground state with $M_S = \pm S$, each corresponding to opposing magnetic orientation along the easy axis. Thus, $M_S = +S$ corresponds

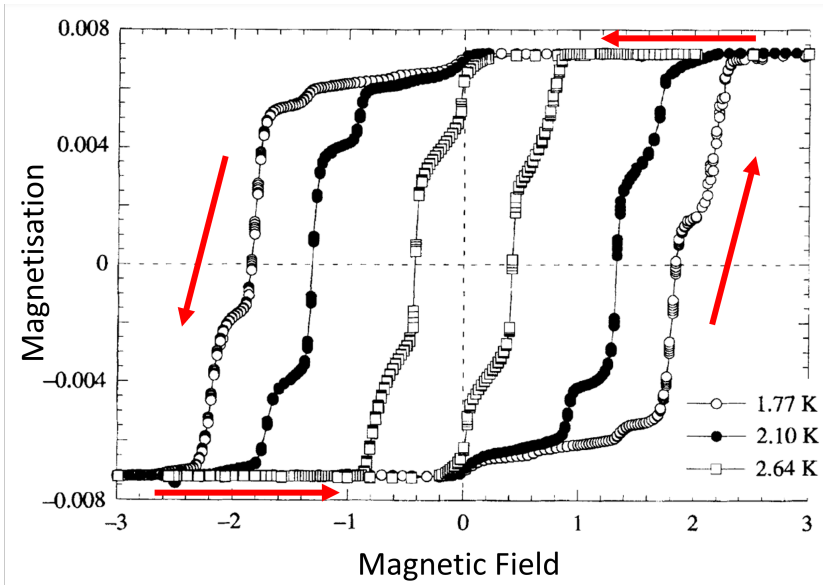


Figure 2.1: Hysteresis loop of Mn_{12}-ac, the first single molecule magnet discovered in 1993, at three different temperatures.[2]

to spin up or 1 in binary, while $M_S = -S$ stands for spin down or 0. For the inverse case, $D > 0$, it can be seen that the ground state corresponds to $M_S = 0$ and no bistable ground state, and hence, no information storage is possible. This highlights that SMMs require a negative anisotropy parameter. In particular, a highly negative anisotropy is desired in order to increase the energy barrier, and equivalently T_B, as much as possible to avoid relaxation back to equilibrium. Figure 2.2 illustrates the bistable ground state present for $D < 0$ and how information is stored in SMMs.

In reality, the relaxation mechanisms are far more complex than solely overcoming the energy barrier U, and are extremely sensitive to numerous factors. The most common relaxation process occurs through spin-phonon coupling and the relaxation rate is given by the below equation,

$$\tau^{-1} = \tau_0^{-1} \exp\left[\frac{-U}{k_B T}\right] \tag{2.1}$$

where τ_0 is known as the *attempt frequency* and can be thought of as the spin-phonon collision rate. The goal is to try and minimise this rate for fixed high T, thus U must be maximised through finding an SMM with the most negative anisotropy parameter D. This statement forms the basis of our goal for this project, to explore a large number of molecules to find the best SMMs with highly negative D.

The primary issue in exploring the chemical space for SMM candidates is that computation of the anisotropy parameter D of each molecule is very expensive. Precise calculations of D require high-throughput computing with a combination of density functional theory (DFT) and full wavefunction calculations, and can take in the region of 10 hours computing time for a molecule of modest size. Clearly this approach is not feasible to efficiently explore large regions of the chemical space. A more intelligent approach must be adopted, leveraging the power of machine learning. In particular, for this project the reinforcement learning paradigm of machine learning was implemented, as discussed in the following section.

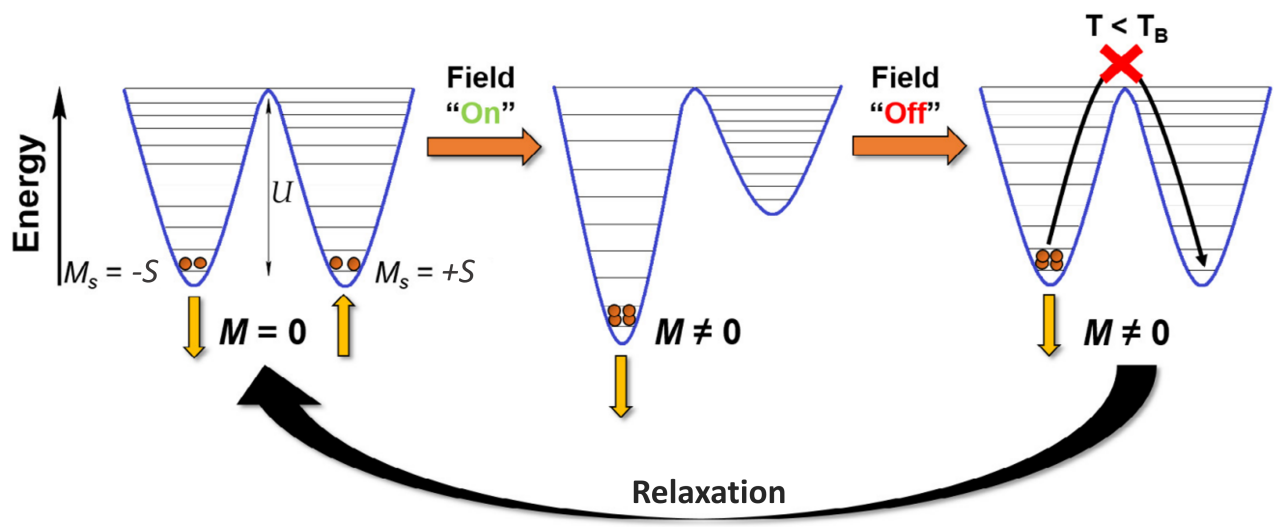


Figure 2.2: Magnetization projections (yellow arrows) of an SMM in equilibrium in absence of external magnetic field with net 0 magnetization (left), magnetization of SMM stabilized in energetically favourable direction upon the application of external magnetic field (center) and subsequent blocking of the magnetization upon the release of magnetic field for $T < T_B$.[3]

2.2 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm that can be summarized as the process of learning what to do, i.e. mapping situations to actions, in order to maximise a numerical reward signal. There is no supervisor, the optimal action to take is never explicitly stated, it must be uncovered through trial and error. Feedback is delayed, in that actions may not only affect the immediate reward but also future situations and consequently all subsequent rewards. An agent[1] sequentially makes decisions on what actions to take and this, in turn, affects the subsequent data it receives from the environment. Its goal is to select actions to maximise the total future reward.

Formally, RL is based on the *reward hypothesis*. A reward R_t is a scalar feedback signal that indicates how well the agent is performing at time step t. The reward hypothesis states that all goals can be described by the maximisation of expected cumulative reward.

Consider the interaction between the agent and environment at each time step t. The agent executes action A_t received by the environment and the environment emits observation O_t and reward R_t, both received by the agent. This sequence of observations, actions and rewards is known as the *history* $H_t = A_1, O_1, R_1, \ldots, A_t, O_t, R_t$ and determines what happens next. A more concise way to capture the useful information contained in the history is in the representation of state.

Formally, *state* is a function of the history: $S_t = f(H_t)$. If the state contains all useful information from the history that makes a difference to the future it is said to have the Markov property.
A state S_t is Markov if and only if $\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t]$, that is given the present state the future is independent of the past.

Further subelements of a RL system include a policy and a value function. A *policy* π defines the agent's behaviour at a given time. It is a mapping from state to action and in general, policies may be stochastic, specifying probabilities to take each action. The goal is to find an optimal policy π_* that maximises expected cumulative reward.

A *value function* gives a prediction of future reward. Different to the reward signal which is an

[1] An agent is defined as an entity that perceives and acts upon its environment comprising of everything outside the agent.

indication of what is good in an immediate sense, the value function specifies what is good in the long run. The value function is used to evaluate states and returns the total amount of reward an agent can expect to accumulate over the future, starting in that state. The value function is dependent on the policy that the agent follows and is a useful tool to evaluate the performance of a policy.

Another important consideration in RL is the balance between exploration and exploitation. Exploitation involves taking greedy actions that exploit current knowledge to select the action whose estimated value is the greatest. However, it is also essential to expand this knowledge by exploring more of the environment, which may result in a greater total reward in the long run.

2.2.1 Markov Decision Processes

Markov decision processes (MDPs) are a classical formalization of sequential decision-making problems and provide a mathematically idealized form of the RL problem.

The interaction between agent and environment is shown in Figure 2.3.

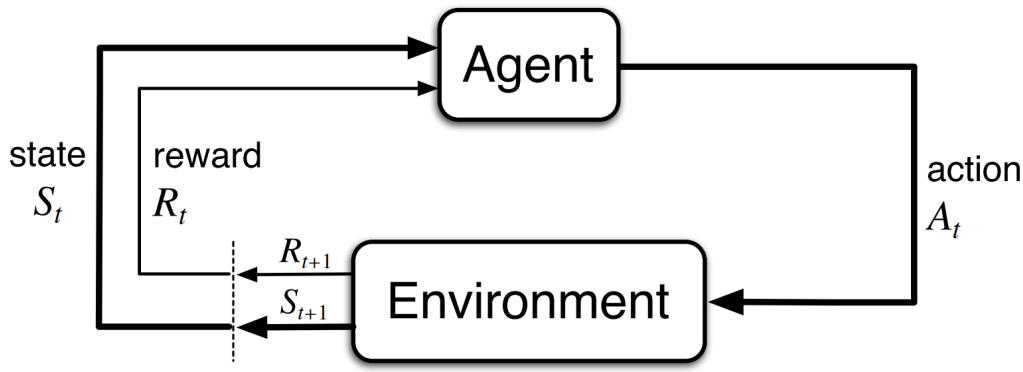


Figure 2.3: The agent-environment interface in a Markov decision process.[4]

At each time step t the agent receives some representation of the environment's state, $S_t \in \mathcal{S}$, where $\mathcal{S}$ is a set of possible states which all possess the Markov property. On the basis of this state, the agent selects an action $A_t \in \mathcal{A}(S_t)$ where $\mathcal{A}$ is a set of possible actions that, in general, may depend on the current state. The agent then receives a numerical reward, $R_{t+1} \in \mathbb{R}$, in part as a consequence of its action. The environment may change due to the action and the agent finds itself in a new state S_{t+1}, giving rise to a sequence of states, actions and rewards. This sequence is known as a *trajectory* of an MDP.

In the case of this project, only *finite* Markov decision processes are considered for which the sets $\mathcal{S}$ and $\mathcal{A}$ have a finite number of elements.

Importantly, MDPs are also equipped with a state transition probability matrix $\mathcal{P}$ and a reward function $\mathcal{R}$. The probability to transition from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ through the selection of action $a \in \mathcal{A}(s)$ is written as in (2.2). The expectation value of the reward due to selecting action a in state s is written as in (2.3).

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s,\ A_t = a] \tag{2.2}$$

$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s,\ A_t = a] \tag{2.3}$$

Another useful quantity to consider is a function of the reward sequence called the return G_t. The return is the total discounted reward from time-step t onwards:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.4}$$

$$\implies G_t = R_{t+1} + \gamma G_{t+1} \tag{2.5}$$

where $\gamma \in [0, 1]$ is the user defined discount parameter that progressively decreases the weight of future rewards. This is used for stability and convergence to avoid infinite returns, and also represents uncertainty looking towards the future. The goal is to now maximise the expected return. Note that the recursive property of the return, given by (2.5), is an important property for the theory and algorithms of RL.

A policy is a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t, then $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$ is the probability that $A_t = a$ is selected if $S_t = s$.

Two useful value functions can be defined under a policy π. The *state-value* function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2.6}$$

The *action-value* function $q_\pi(s, a)$ of an MDP is the expected return starting from state s, taking action a and thereafter following policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{2.7}$$

From the recursive property of the return given in (2.5), the value functions can similarly be decomposed into a combination of immediate reward plus discounted value of successor state (or state-action pair):

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \tag{2.8}$$

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \tag{2.9}$$

It is straightforward to relate the *state-value* and *action-value* functions by taking a 'look ahead' from a state to its possible successor states. Consider a transition from state s, to state-action pair (s, a), to new state s'. A useful tool to visualise these transition dynamics are backup diagrams. States are represented by open circles while state-action pairs are represented by smaller solid circles. Figure 2.4 shows backup diagrams for the transitions mentioned above.

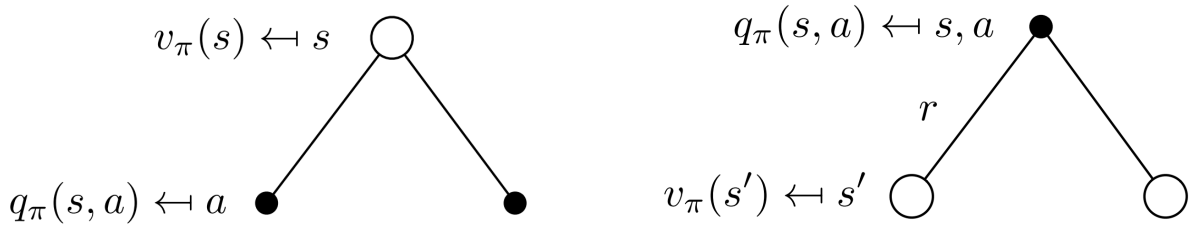


Figure 2.4: Backup diagram for transitions from state to state-action pair (left) and state-action pair to successive state (right).[5]

It can be seen that each backup diagram leads respectively to an equation below to relate the *state-value* and *action-value* functions:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) \tag{2.10}$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \tag{2.11}$$

Equation (2.10) can be interpreted as the state value being equivalent to the average of the action values weighted by the probability of taking each action according to policy π.

Equation (2.11) can be interpreted as the action value being equivalent to the expectation of the immediate reward plus the discounted average of the subsequent state values, weighted by the state transition probabilities.

Combining the backup diagrams in Figure 2.4, or equivalently, combining equations (2.10) and (2.11) leads directly to the *Bellman expectation equations* for v_π and q_π:

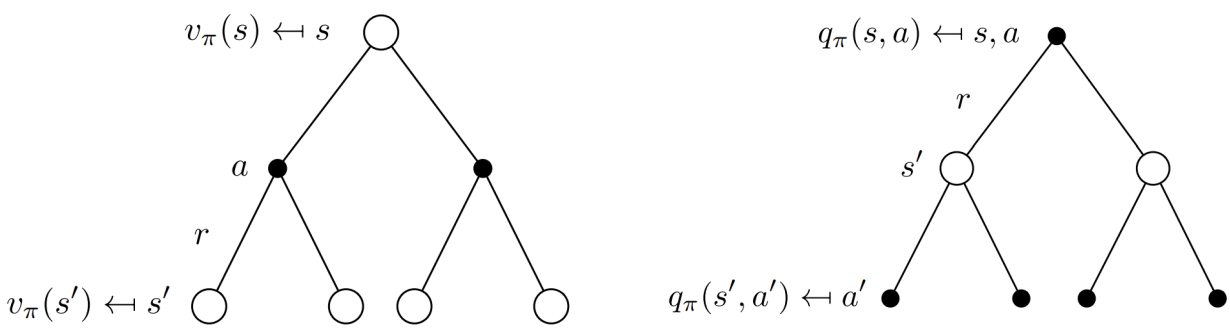


Figure 2.5: Backup diagram for v_π (left) and q_π (right).[5]

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \Big(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \Big) \tag{2.12}$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a') \tag{2.13}$$

The Bellman expectation equation gives a description of the system that can be solved for the value function. Formally, the state transition dynamics and reward function can be averaged over policy π, forming a Markov reward process (MRP). The Bellman equation for MRPs is a linear equation that can be solved directly for the value function, given the problem is small enough. In reality more efficient iterative methods are used to solve larger MRPs.

As previously stated, the goal in RL is to find an *optimal policy* π_* that maximises the value functions. A partial ordering over policies must be defined to compare the performance of difference policies. A policy π is defined to be an 'improvement' over policy π' by the partial ordering:

$$\pi \geq \pi' \ \textit{iff}\ v_\pi(s) \geq v_{\pi'}(s),\ \forall\ s \in \mathcal{S} \tag{2.14}$$

For any MDP there always exists at least one optimal policy π_* such that, $\pi_* \geq \pi,\ \forall\ \pi$. The state value and action value functions following an optimal policy are defined as the *optimal state-value function* and *optimal action-value function*:

$$v_{\pi_*}(s) = v_*(s) = \max_\pi v_\pi(s) \tag{2.15}$$

$$q_{\pi_*}(s,a) = q_*(s,a) = \max_\pi q_\pi(s,a) \tag{2.16}$$

In addition, there always exists a deterministic optimal policy that can simply be found by maximising over $q_*(s,a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a\in\mathcal{A}}{\text{argmax}}\ q_*(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

such that if $q_*(s,a)$ is known, so too is the optimal policy π_* and the problem is solved.

Similar to the derivation of the Bellman expectation equation, the optimal value functions can be related to themselves via a 'look ahead'. The backup diagrams for the optimal value functions now feature arcs at the action choice points to represent that the maximum over that choice is taken, rather than the expected value following some policy:

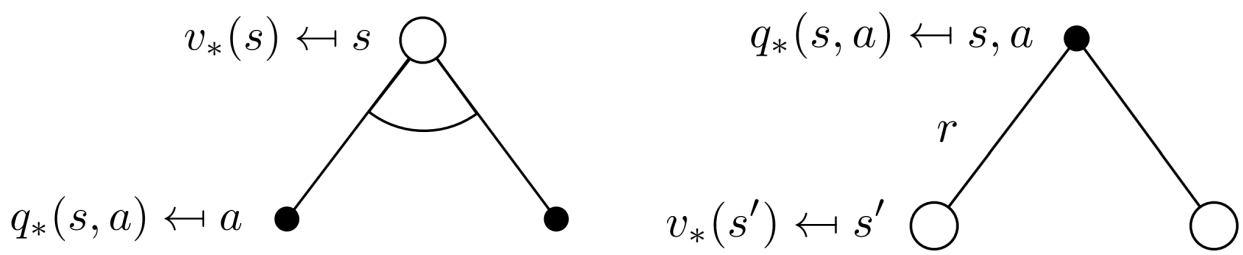


Figure 2.6: Backup diagrams for optimal value functions.[5]

This leads to the below equations:

$$v_*(s) = \max_{a\in\mathcal{A}} q_*(s,a) \tag{2.18}$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \tag{2.19}$$

As before, combining the backup diagrams, or equivalently, equations (2.18) and (2.19) leads to the *Bellman optimality equations* for v_* and q_*:

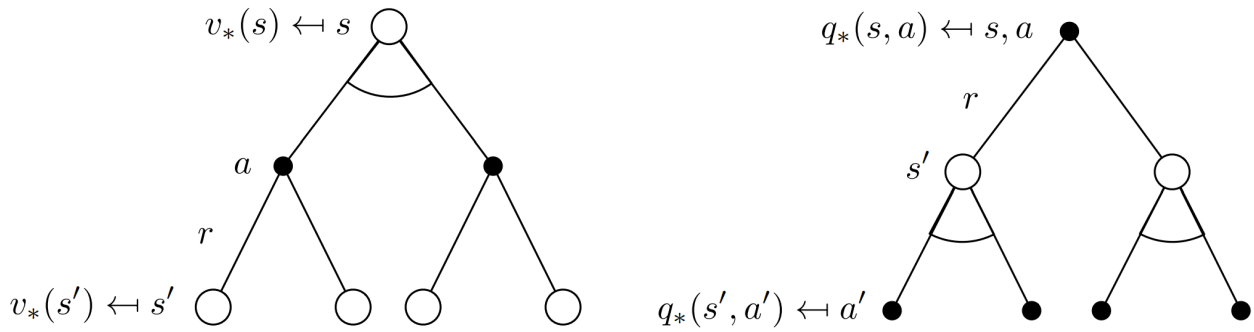


Figure 2.7: Backup diagram for v_* (left) and q_* (right).[5]

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \tag{2.20}$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a') \tag{2.21}$$

It is clear that the Bellman optimality equation is non-linear, so in general there is no closed form solution. Iterative solution methods must be implemented such as dynamic programming, Monte Carlo methods or temporal difference learning methods. In particular for this project, a common temporal difference learning algorithm known as Q-learning was implemented, as discussed in the following section.

2.2.2 Q-Learning

As previously mentioned, finding the optimal policy π_* of an RL problem is equivalent to solving for the optimal action-value function $q_*(s,a)$ (see 2.17). Thus, to solve an RL problem the Bellman optimality equation (2.21) must be solved for $q_*(s,a)$. In this project a well-known method to solve for $q_*(s,a)$ was implemented, known as Q-learning.

Q-learning is an off-policy temporal difference learning algorithm in which the learned approximation of the action-value function, Q, converges to q_*.

Temporal difference learning involves *bootstrapping* in which estimates are updated before the final outcome is known based on the current approximation of the value function. This is quite different to Monte Carlo methods in which updates are made only after the final outcome is known.

Off-policy refers to the use of two distinct policies: a behaviour policy and a target policy. The behaviour policy is used to generate trajectories of the MDP that the target policy then learns from. In other words, off-policy algorithms improve a policy that is different from the policy used in action selection. In particular for Q-learning, a common behaviour policy known as an $\varepsilon -$ greedy policy is used. Here $\varepsilon \in [0,1]$ refers to the user defined exploration parameter. At each time step the $\varepsilon -$ greedy policy selects a random exploratory action $a \in \mathcal{A}$ with probability ε, or otherwise takes an exploitative action with respect to current knowledge[2] $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s,a)$ with probability $1 - \varepsilon$. The target policy is completely greedy with respect to the Q-value estimates.

[2]Value ties are split arbitrarily.

Determining the optimum exploration parameter ε is a crucial step in solving an RL problem efficiently. The process of determining the optimal ε is largely empirical, requiring trial and error to observe the value that results in the best performance.

Q-learning is a tabular method, in that a table with states as rows and actions as columns, is populated with the current action-value estimates of each state-action pair. The initialization of the Q-table is somewhat arbitrary and is usually set as 0, however $Q(S_{\text{terminal}}, \cdot\,) = 0$ must always be enforced.

The update scheme for the Q-learning algorithm is shown below and is implemented for each step of the $\varepsilon -$ greedy behaviour policy:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big] \tag{2.22}$$

where $\alpha \in (0, 1]$ is the learning rate parameter that determines the extent that newly acquired information overrides old information. It can be seen that this method updates the Q-value towards $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$. This is known as the *TD target* and it can be seen that it closely resembles the Bellman optimality equation for q_* (2.21). This update scheme continues until S is terminal, and is then repeated starting again at an initial S. Each repetition constitutes an *episode* of the MDP and many episodes are repeated until sufficient learning is achieved (i.e. sufficient convergence of the Q-table). As the Q-value estimates improve over time, so too does the $\varepsilon -$ greedy behavior policy and the completely greedy target policy. As long as the Q-value estimates of all state-actions pairs continue to be updated, the Q-learning algorithm has indeed been heuristically shown to always converge to q_*, effectively solving the RL problem.

2.3 Computational Method

As previously stated, the goal of this project is to efficiently screen a large number of molecules for promising SMM candidates. This was achieved through the use of reinforcement learning to learn how to construct molecules with a highly negative magnetic anisotropy (D parameter). In the case of this project a central binding atom of cobalt was used due to its highly degenerate $3d$ orbital and large axial magnetic anisotropy. The high degeneracy in the $3d$ orbital gives significant orbital angular momentum allowing for the construction of high spin molecules with large energy barriers to relaxation.

The Q-learning algorithm was implemented to assemble molecules by 'attaching' ligands[3] one by one to the cobalt atom. Each ligand addition is an action in the MDP determined through the $\varepsilon -$ greedy behaviour policy. A reward of 0 is received after each action and the Q-table is updated accordingly at each step. The new state is represented by the entire construction so far such that it possesses the Markov property. This continues until the last ligand is added and the final molecule is formed. The anisotropy D is then calculated for the molecule and a reward of $-D$ is passed to the RL agent, as we want to maximise for negative D. This constitutes one episode of the MDP. The process is repeated for many episodes, each initialised with no ligands attached, until sufficient learning is achieved.

In this way we hope that a pattern can be learned regarding which ligand combinations are generally more desirable or less desirable in achieving molecules with highly negative D, and thus discover SMMs that function effectively at higher ambient temperature.

It should be mentioned that in practise this project used datasets with precomputed anisotropy values for large sets of molecules, constructed from a set of possible ligands that the RL agent had at its disposal. This simplifies the investigation of the effectiveness of the Q-learning algorithm for this problem, as the expensive computation of D is not required after assembling each molecule.

[3] A ligand is a molecule or ion with a functional group that binds to a central metal atom to form a coordination complex.

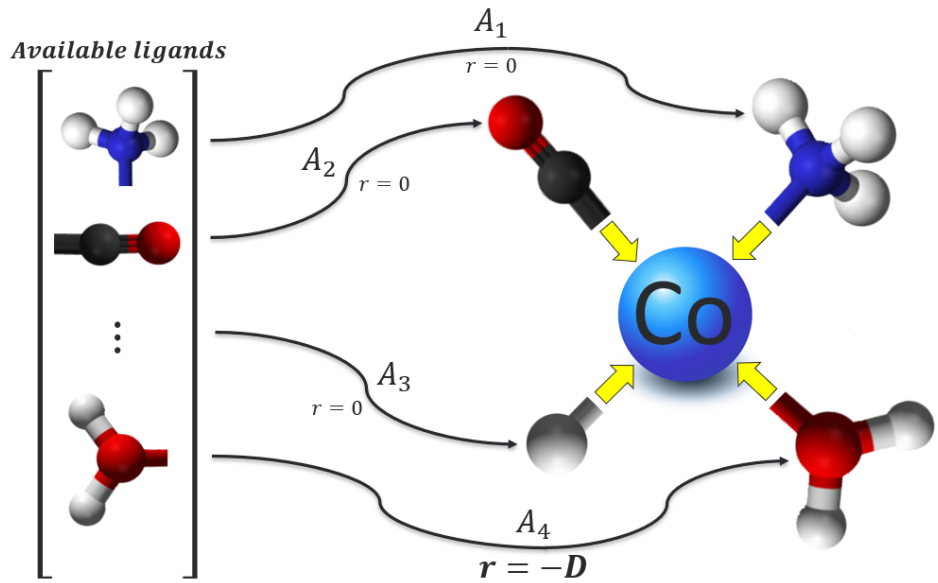


Figure 2.8: Example episode of the MDP. Ligands are selected from the available set to bind with the central binding atom of cobalt. No reward is received until the final step when $r = -D$, as the goal is to maximise for negative anisotropy.

The number of molecules that must be constructed before the Q-learning algorithm finds the 'best' molecule with the most negative D, compared with the total possible molecules that could be assembled from the set of ligands, gives an indication of how much of the chemical space had to be explored before finding the optimum molecule. The hope is that the Q-learning algorithm efficiently discovers the best SMM candidates without exploring a large area of the chemical space, as in practise this would drastically reduce the number of expensive D calculations required to find promising SMMs.

It should be mentioned that this is an extremely unusual and challenging type of RL problem. Each episode of the MDP is very short and the agent receives no non-zero reward until the final step. Typically, RL problems consist of many more MDP steps with information being provided to the agent throughout the MDP with non-zero intermediate rewards.

Another consideration due to this particular form of MDP is that the learning rate parameter α and the discounting parameter γ can both be set to 1 in the Q-learning update scheme (2.22). This is a consequence of only receiving rewards of zero until the final step, deterministic state transitions[4] and the knowledge that every episode terminates. This results in a simplification of the Q-learning update scheme (2.22):

$$Q(S_t, A_t) \leftarrow R_{t+1} + \max_a Q(S_{t+1}, a) \tag{2.23}$$

such that an intermediate state-action pair updates to the max Q-value of being in the new state and a state-action pair transitioning to a final molecule state simply updates to the $-D$ reward. Additionally, the order in which the ligands are added is invariant with regard to the final molecule that is assembled.

In the case of this project, the Q-learning algorithm is being pushed to the limit. Learning anything of value with such a short MDP and with such little information being provided is expected to be challenging, but hopefully not impossible.

[4]State transitions are deterministic in that taking an action of adding a ligand will with probability 1 result in a new state with that ligand attached.

3

Results & Discussion

3.1 Simple Synthetic Data

The initial stage of the project entailed determining if the Q-learning method was an effective approach for even the simplest cases of the type of RL problem described in section 2.3. To investigate this, we used randomly generated data to create synthetic or fake datasets of ligands, molecules and corresponding anisotropy values. In particular, the synthetic anisotropy values were generated by assigning each ligand a random value and then simply summing over the ligand values in the final molecules. If the value of the ith ligand in the molecule is l_i and there are M ligands in the final molecule then the anisotropy is generated by the simple reward function $D(\text{molecule}) = \sum_{i=1}^{M} l_i$. In this case the optimum molecule is trivial, simply being the one with with the 'best' or most negative ligand repeated M times. In other words, the optimum molecule is of the form,

$$\underbrace{[A, A, \ldots, A]}_{M \text{ times}} \text{ where } A \text{ is the ligand with the most negative value.} \tag{3.1}$$

The parameters introduced to control the size of the generated synthetic datasets were the number of ligands available N and the size or number of ligands in the final molecule M. This gives the total number of possible molecules that could be constructed as $n_{\text{poss}} = \binom{M+N-1}{N-1}$, a combinations with replacement expression as repeated ligands are permitted. It also gives the total number of possible states, including intermediate states as $n_{\text{states}} = \sum_{m=0,1,\ldots,M} \binom{m+N-1}{N-1}$ such that the size of the Q-table with states as rows and ligand addition actions as columns is $N \cdot n_{\text{states}}$. It can be seen that the state space size displays a factorial increase growing quickly with increasing N and in particular, increasing M.

To analyse the effectiveness of the Q-learning algorithm two indicator ratios were defined, a *performance* ratio and a *cost* ratio. If n_{mol} is the total number of molecules constructed, i.e. the total number of episodes, and $n_{\text{uniq mol}}$ is the total number of unique molecules constructed then:

$$\text{Performance ratio} = \frac{n_{\text{mol}}}{n_{\text{poss}}}, \qquad \text{Cost ratio} = \frac{n_{\text{uniq mol}}}{n_{\text{poss}}} \tag{3.2}$$

The performance ratio gives an indication of the efficiency of convergence to the optimum molecule. It describes how many episodes of the Q-learning algorithm were required before learning the best molecule relative to the size of the state space of possible molecules. It can also be thought of as the average number of times every molecule in the dataset had to be constructed before achieving convergence to the optimum. Where the performance ratio considers the possibility of repeating the construction of identical molecules, the cost ratio is only concerned with the unique molecules constructed. The cost ratio is bound in the range $[0, 1]$ and gives an indication of the proportion of the chemical state space that had to be explored before discovering the optimum molecule. In other words, it represents the number of D computations required relative to the size of the state space of possible molecules. The goal is to attempt to minimise these ratios as much as possible, however in practise minimising the cost ratio is the main focus as the goal is to reduce the number of D computations required as much as possible. A compromise of high performance ratio with low cost ratio is acceptable and often necessary as the computation time of a slowly converging Q-learning algorithm is negligible compared to increasing the number of anisotropy computations required.

A synthetic dataset was generated for an initialisation of the parameters $N = 24$ possible ligands and $M = 4$ final molecule size with each random ligand value in the range $[-10, 10]$. This gives $n_{\text{states}} = 20,475$ and $n_{\text{poss}} = 17,550$. The exploration parameter was initially set to $\varepsilon = 0.5$ such that the probability of taking exploratory and exploitative actions was equal. 200 iterations of the Q-learning algorithm were performed to convergence with the average of such a run plotted in Figure 3.1. The maximum Q-value entry in the Q-table, in other words the current best $-D$ molecule value is plotted against increasing number of episodes.

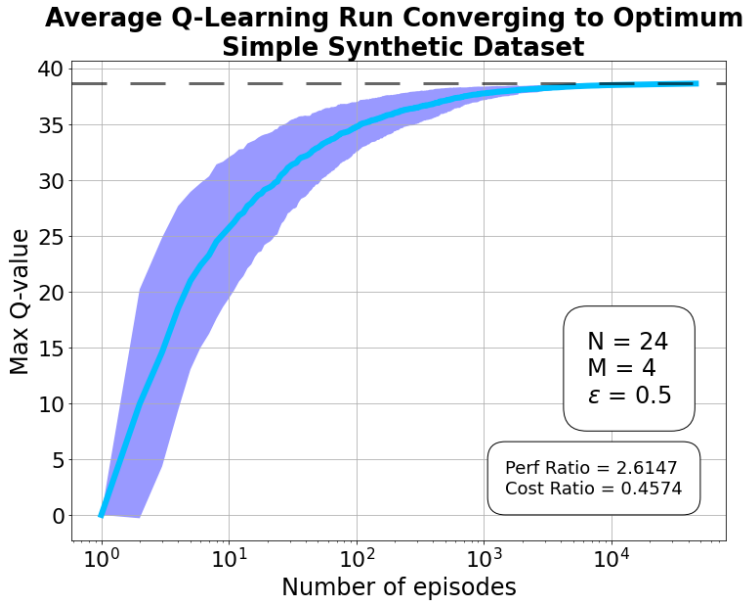


Figure 3.1: Semi-log plot of the maximum Q-value (within 1 standard deviation) vs no. episodes for an average Q-learning iteration of an initialised $N = 24$, $M = 4$, synthetic dataset with $\varepsilon = 0.5$ and performance and cost ratios shown.

It can be seen that the algorithm very quickly discovers some 'good' molecules with high Q-values, however it takes a very large number of episodes to find the absolute optimum molecule. It can be seen from the performance ratio that the total number of molecules constructed was 2.6147 times the number of possible molecules. This indicates that the construction of some molecules was repeated multiple times, which of course is undesirable as new information is gained only upon the first construction of a molecule. It can be seen from the cost ratio that approximately 45.74% of the state space of possible molecules was explored before finding the optimum molecule. This is of course an improvement over randomly sampling the molecules in the dataset, which would result in a cost ratio of 0.5, however it is not a particularly impressive result.

Initial testing with some different ε values suggested that the cost ratio decreased with decreasing epsilon, i.e. a bias towards exploitative action selection. This fits with intuition as the exploration being increasingly restricted should reduce the number of 'unnecessary' molecule constructions, and accordingly the number of D parameter anisotropy computations required.

The optimal exploration parameter ε was investigated by performing a large number of iterations of the Q-learning algorithm for a range of ε values. The resulting performance and cost ratios with their standard deviations were plotted against these ε values in Figure 3.2.

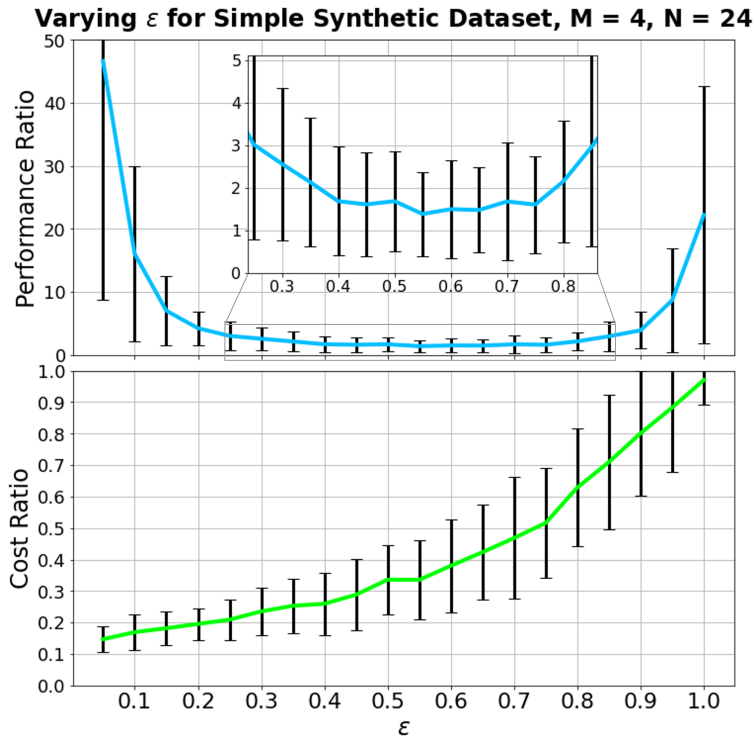


Figure 3.2: Performance and cost ratio vs ε for a synthetic dataset initialised with $N = 24$ and $M = 4$.

Interpreting these results, it can be seen that at very low and very high ε or equivalently for highly exploitative or highly exploratory action selection, the performance ratio is very large, indicating a slow convergence rate. In fact, it can be shown that in the limit $\varepsilon \to 0$, the performance ratio $\to \infty$. An exploration parameter of 0 allows only exploitative actions resulting in the construction of the first discovered positive value molecule to be infinitely repeated and no progress to be made (unless the first molecule is optimal). An exploration parameter of 1 allows only random action selection, which as expected is not an intelligent approach and results in slow convergence to the optimal molecule. It should also be noted that the performance ratio has large variance, in particular for low and high ε. Action selection via the $\varepsilon -$ greedy policy is a stochastic process. This can lead to high volatility in the results of the indicator ratios at each iteration as sometimes a run can be 'lucky' or 'unlucky' in selecting actions that add ligands present in the optimal molecule. High and low ε iterations are more sensitive to this, while a balance between exploration and exploitation is more stable. It can be seen that the range $\varepsilon \in [0.4, 0.75]$ results in the lowest performance ratio with a minimum of 1.3856 ± 0.9867 at $\varepsilon = 0.55$, corresponding to the highest efficiency in convergence. This is the optimal ε in regard to the performance ratio.

Observing the results for the cost ratio, it can be seen that there is a general increase with increasing ε, with the variance also increasing with ε. This is to be expected as increasing ε loosens the restriction on exploration resulting in a higher number of unique molecules assembled and a higher sensitivity to the stochastic nature of the process. It can be seen that the cost ratio is highest for $\varepsilon = 1$, completely random action selection. This is expected as an unintelligent approach is sure to require the construction of the highest number of 'unnecessary' molecules. The cost ratio was found to be 0.9701 ± 0.0785 such that it required the construction of 97.01% of the possible molecules to find the optimal one. This was initially an unexpected result, as it is worse than the case of randomly sampling molecules from the dataset, resulting in a cost ratio of 0.5. However if the simple form of the optimal molecule 3.1 is considered, it becomes clear that the same ligand addition action needs to be selected at *every* step and this restricts the number

of trajectories reaching the optimal molecule to just 1. The probability of randomly selecting the correct ligand at each step is $\frac{1}{N} = \frac{1}{24}$, and each iteration consists of $M = 4$ steps giving $P(\text{optimal}) = \frac{1}{N^M} = \frac{1}{24^4}$ for each iteration. Thus, in $24^4 = 331,776$ iterations one occurrence of the optimal molecule is expected, and this agrees within variance to the $358,324$ required to generate the $\varepsilon = 1$ datapoint.

The minimum value of the cost ratio was 0.1470 ± 0.0406, occurring at the minimum ε value of 0.05 such that only $14.70 \pm 4.06\%$ of the possible molecules required assembly before the optimal molecule was found. This minimum at low ε is expected as exploration is highly restricted and the little exploration that does occur creates limited deviations in the most promising trajectory. Of course, the cost ratio will decrease further with even smaller ε, however only computationally convenient ε values are considered for demonstration purposes. As previously mentioned, the necessary compromise between low cost ratio and high performance ratio is evident from these results. It is worth restating that the benefit of reducing the number of required D computations far outweighs the drawback of increased performance ratio. It can be concluded that for this most simple case, the Q-learning algorithm has indeed successfully learned the optimal molecule in the dataset while restricting the number of unique molecules that had to be constructed.

The effect of changing the parameters N and M was also investigated for a fixed $\varepsilon = 0.5$ exploration parameter. For each initialisation of N and M, a new synthetic dataset was generated, and a number of Q-learning iterations were performed to give an approximation of the performance and cost ratio associated with each dataset. The results for each indicator ratio are shown in separate heatmap plots:

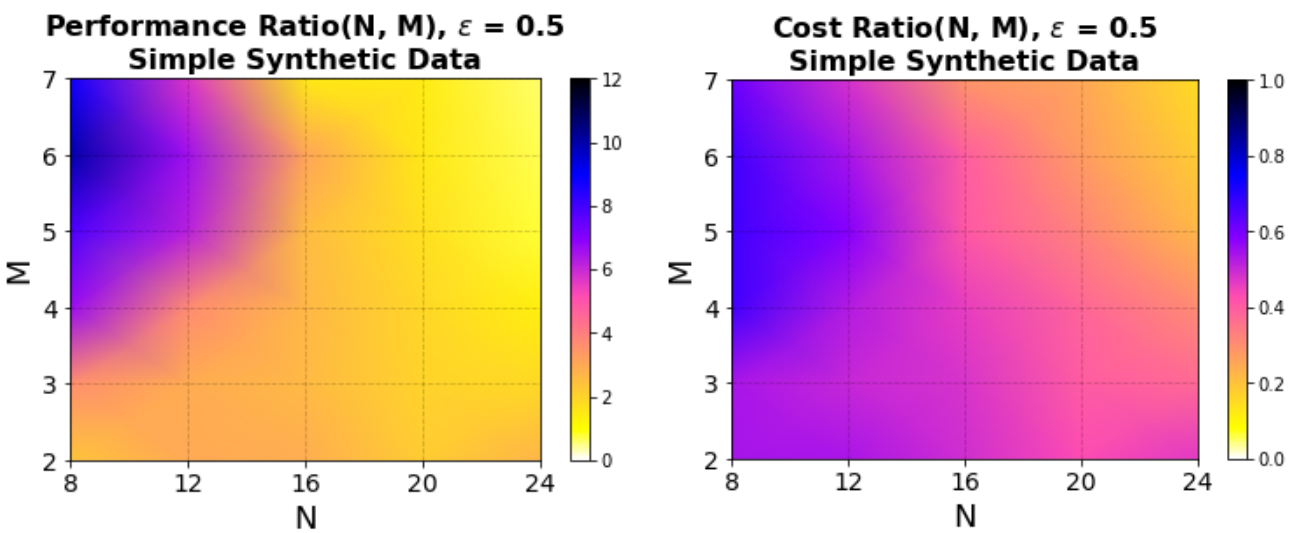


Figure 3.3: Plots showing the effect of changing the number of ligands available N and size of the final molecule M on the performance ratio (left) and the cost ratio (right) for fixed $\varepsilon = 0.5$.

It can be seen that in general the performance ratio and cost ratio both decrease for increasing N and M. This is to be expected as the Q-learning algorithm is more effective for bigger MDPs. Of course, the number of molecules (and unique molecules) that have to be assembled increases with both N and M but this increase is small relative to the factorial growth of the state space of possible molecules. Interestingly, it can be seen that for small N the indicator ratios actually increase with increasing M. This is because at N and M comparable, the rate of increase of n_{poss} is not as sensitive to increases in N and M when compared to the rate of change when N and M are very different (see A.3 for derivation). In this case n_{poss} grows slowly compared to the increasing number of molecules that have to be constructed.

3.1.1 Q-learning Improvement

It was realised early in the project, that due to the order invariance when adding ligands, an improvement to the update scheme of the Q-learning algorithm could be implemented. In particular, the order invariance allows for multiple updates of the Q-table as many different intermediate states can lead to the same final molecule. Figure 3.4 outlines an example of the improved update scheme for the final step of an episode of a three ligand molecule MDP. In this case a transition from state $[A, B]$ to terminal state $[A, B, C]$ through action C is considered.

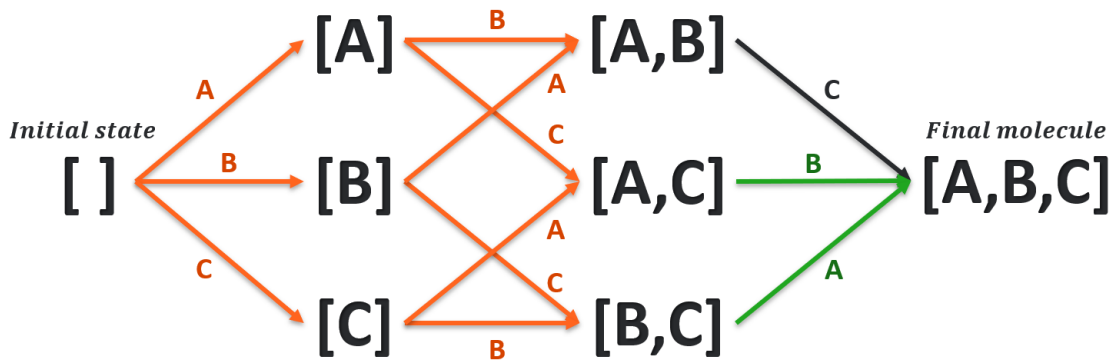


Figure 3.4: Schematic of the improved Q-learning scheme with multiple state-action pair updates.

Under normal circumstances, state $[A, B]$ and action C is the only state-action pair that is updated in the Q-table via (2.23) shown with the black arrow. However, it is clear that states $[A, C]$ and $[B, C]$ also reach the same final molecule $[A, B, C]$ through actions B and A respectively. These state-action pairs can also be updated with the same Q-value, shown with green arrows. Additionally, all intermediate state-action pairs that could have reached the same final molecule can be considered for the same update. However, in this case it is possible that the state-action pairs could reach different final molecules, for example through the later addition of ligand D. This means that these state-action pairs may already have a Q-value in the Q-table better than if they were to reach final molecule $[A, B, C]$. As a result, it is not possible to blindly overwrite the current Q-value of these state-action pairs as it is required that updates always improve the Q-value approximation. A maximum must be taken over the current Q-value and the new Q-value being considered for update for each of the intermediate state-action pairs represented by the orange arrows.

A consequence of this maximum is that the Q-table can no longer be initialised to 0, rather it must be initialised with value of the 'worst' possible molecule in the dataset. This is because otherwise it is possible for the maximum to select the initialised 0 value rather than a lower value, and learning information about 'bad' molecules with value $Q < 0$ is prevented.

Naturally, this improved update scheme results in a significant improvement in the efficiency of populating the Q-table, with the improvement being even more dramatic for larger molecule MDPs.

The above improvement to the Q-learning algorithm was implemented and the analysis performed in section 3.1 was repeated for the same datasets (See appendix A.1 for average single iteration plot):

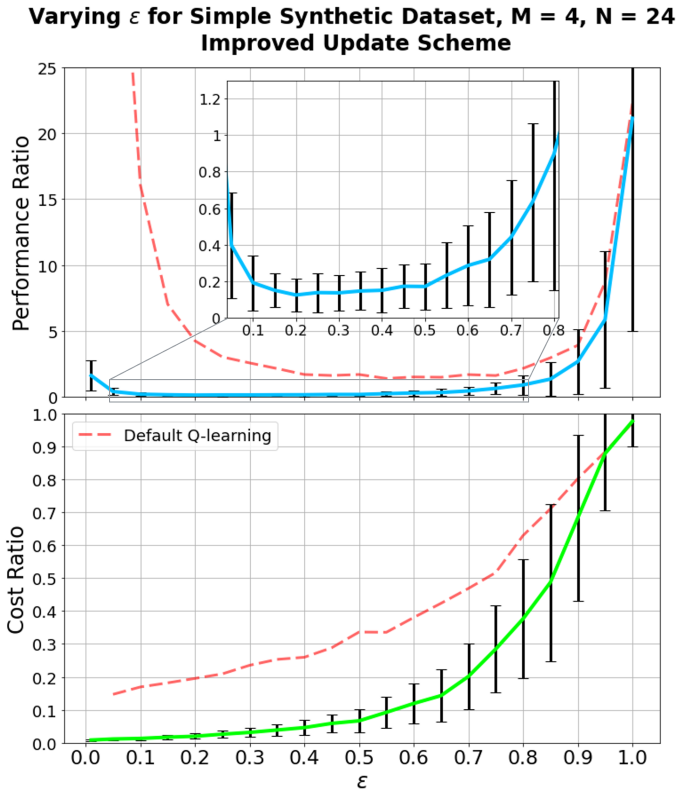


Figure 3.5: Indicator ratios vs ε with improved Q-learning update scheme implemented for the same dataset as in Figure 3.2 (results prior to the implementation of improvement shown in red).

It can be seen that the implementation of this improved update scheme significantly improves the performance of the Q-learning algorithm when compared to the results prior to this implementation. As $\varepsilon \to 1$ the two results converge to approximately the same value in both the performance ratio and cost ratio. This is expected as exploitative actions are required to benefit from the improved information flow provided by the updated scheme. As ε decreases, a significant improvement in both ratios is observed. In regard to the performance ratio the minimum value is shown to be 0.1240 ± 0.0899, achieved at $\varepsilon = 0.2$, with similar values achieved in the range $\varepsilon \in [0.1, 0.5]$. This indicates that a bias towards exploitative actions is now favoured, compared to a slight bias towards exploration being favoured in the unimproved case.

As $\varepsilon \to 0$ a huge improvement is observed in the performance ratio when compared to the diverging result of default Q-learning case. This improvement can be explained through understanding how, under this improved scheme, the number of exploratory actions required to find the optimum molecule is very low. This is a consequence of value ties being split arbitrarily when taking exploitative actions. Consider the first episode of the Q-learning algorithm in which one of the ligands in the final molecule is optimal. The improved scheme sets the entire update tree (as in Figure 3.4 equivalent) to the same Q-value resulting in a high number value ties allowing for 'passive' exploration when taking exploitative actions. As a result, relatively few $\varepsilon-$greedy exploratory actions are required to populate a large number of high-value regions of the Q-table with optimal ligand contributions. This leads to a higher rate of convergence to the complete optimum molecule. Despite this, it should still be stated that $\lim_{\varepsilon \to 0} \big(\text{Perf. Ratio}\big) = \infty$ holds regardless of the improvement.

In regard to the cost ratio, the minimum value is shown to be 0.0084 ± 0.0023, attained at the lowest investigated value of the exploration parameter $\varepsilon = 0.01$. This indicates that only $0.84 \pm 0.23\%$ of the

possible molecules had to be constructed before the optimal molecule was found. This is an encouraging result, however the optimum molecule is the trivial case 3.1, and the Q-learning algorithm is expected to perform very well for such a simple MDP.

The effect of varying N and M was again investigated for the improved update scheme:

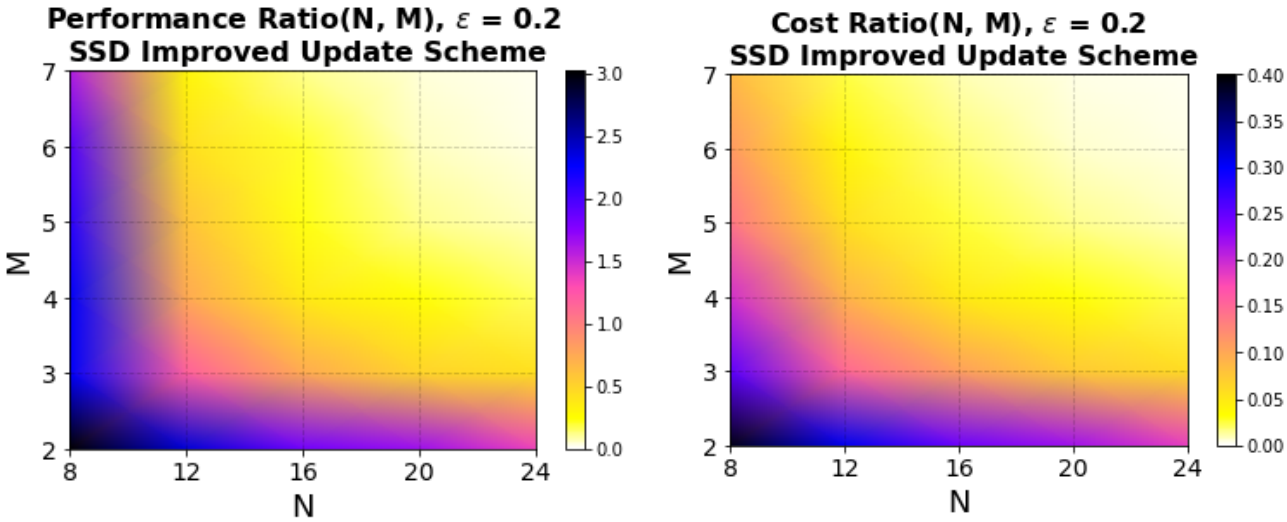


Figure 3.6: Effect of varying N and M on the performance ratio (left) and cost ratio (right) for fixed $\varepsilon = 0.2$, using generated simple synthetic datasets (SSD) with the improved Q-learning update scheme implemented.

It can be seen that both the performance ratio and cost ratio decrease quickly with increasing N and in particular increasing M. This is again due to the rate of increase in the number molecules requiring construction being much smaller than the rate of increase of n_{poss} with increasing N and M. As the size of problem grows, the results become further improved.

3.2 Complex Synthetic Data

The next stage of the project involved altering the reward function to generate a more complex synthetic dataset in which the optimum molecule was not the trivial case. A number of approaches were attempted to achieve this. One approach was to generate the synthetic anisotropy data through the use of a test function. Test functions are artificial landscapes that are designed specifically to evaluate the performance of optimization algorithms, usually containing noise and multiple local extrema. Each ligand can be assigned a random value in the input domain of the test function such that the generated value of a molecule is the test function evaluated at its ligand input values. In particular, an n-dimensional, symmetric test function is required as the molecule size M is variable and the order of ligand addition is invariant. The Schwefel test function[6] was implemented to generate a synthetic dataset, however, its global extremum occurs on the diagonal which again resulted in the trivial case optimum molecule. Consequently, analysis showed similar results to that of the simple synthetic data (see appendix A.2). The test function requirements of n-dimensionality, symmetry, and global extremum occurring off the diagonal are very restrictive. It transpired that it was not possible to find a test function that satisfied these criteria, so another approach was considered.

This new approach attempted to directly model artificial interaction between ligands in the molecule. Each ligand was assigned a vector of length n with random entries such that the ligand in position l of the final molecule is represented by $\vec{v}_l$. Each pair of ligands in the final molecule were considered to interact

giving $\binom{M}{2} = \frac{M(M-1)}{2}$ interaction terms for a final molecule of size M. This many random symmetric, full rank $n \times n$ matrices $\mathbf{A}_i$ were generated such that the value of the interaction between the first pair of ligands in position 1 and 2 was given by $\vec{v}_1^T \mathbf{A}_1 \vec{v}_2$. Then, for the example of a size $M = 4$ final molecule, the total value of the molecule is given by $\vec{v}_1^T \mathbf{A}_1 \vec{v}_2 + \vec{v}_1^T \mathbf{A}_2 \vec{v}_3 + \vec{v}_1^T \mathbf{A}_3 \vec{v}_4 + \vec{v}_2^T \mathbf{A}_4 \vec{v}_3 + \vec{v}_2^T \mathbf{A}_5 \vec{v}_4 + \vec{v}_3^T \mathbf{A}_6 \vec{v}_4$. With this approach it is now possible for the optimum molecule to be more complex with a combination of different ligands. The form of optimum molecule in the generated dataset is dependent on the random initialisation of ligand vectors and interaction matrices. In the following presentation of results, the matrix/vector dimension was set to $n = 3$ and the random number generation was seeded for each synthetic dataset generated for different M and N parameters, such that the optimum molecule was sufficiently complex i.e. minimal repeated ligands in the optimum molecule.

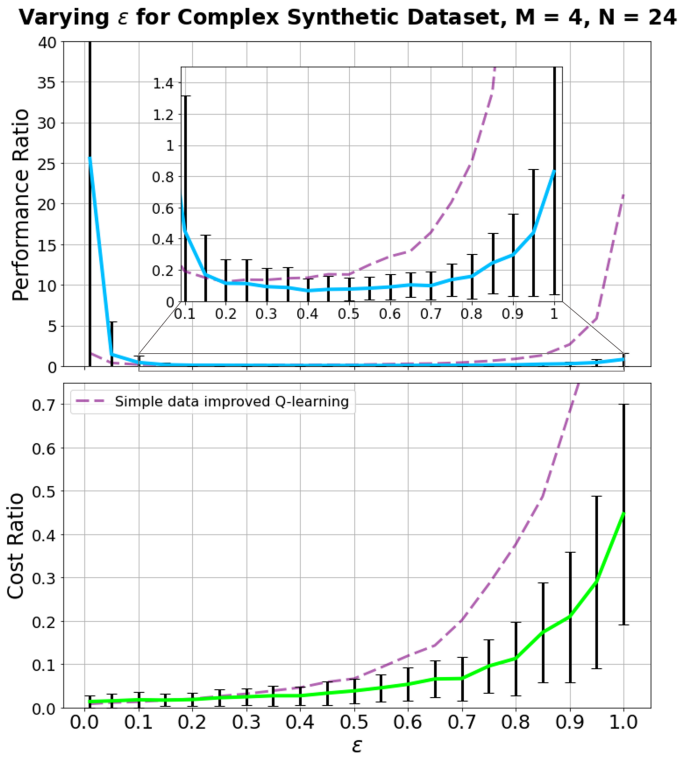


Figure 3.7: Indicator ratios vs ε for an $N = 24$ and $M = 4$ generated complex synthetic dataset. Results are compared to the simple synthetic dataset case in Figure 3.5 (purple).

It can be seen that at high ε the results for the complex dataset are, in fact, an improvement over the results for the simple dataset. This is because in the simple dataset case there is only one trajectory to the trivial optimum molecule with repeated ligands, however, in the complex dataset case there are $M!$ trajectories to the optimum molecule (if there are no repeated ligands in the optimal molecule, otherwise $\frac{M!}{r!}$ trajectories for r repeated ligands). For $\varepsilon = 1$ random action selection, the probability of finding the optimal molecule in each episode is now $P(\text{optimal}) = \frac{M!}{N^M} = \frac{1}{24^3}$ for the case of Figure 3.7.

The minimum performance ratio was found to be 0.0668 ± 0.0751 at $\varepsilon = 0.4$, with similar values in the range $\varepsilon \in [0.2, 0.7]$. This indicates that a close to equal balance of exploration and exploitation is the best approach in terms of convergence rate, with stability in respect to moderate biases in either direction. The maximum performance ratio was again found to occur at the smallest value of $\varepsilon = 0.01$, however it was significantly greater than the simple synthetic dataset case. This highlights the Q-learning algorithm having more difficulty in determining the non-trivial optimal molecule.

The minimum cost ratio was 0.0138 ± 0.0139 at $\varepsilon = 0.01$, slightly higher than the 0.0084 ± 0.0023 result in the case of the simple dataset, which is to be expected. This is a very encouraging result as only having to explore $1.38 \pm 1.39\%$ of the state space of possible molecules represents a huge reduction in the number of D anisotropy computations required to find the optimal molecule.

The effect of varying N and M was investigated for complex synthetic datasets, seeded such that the optimum molecule was sufficiently complex:

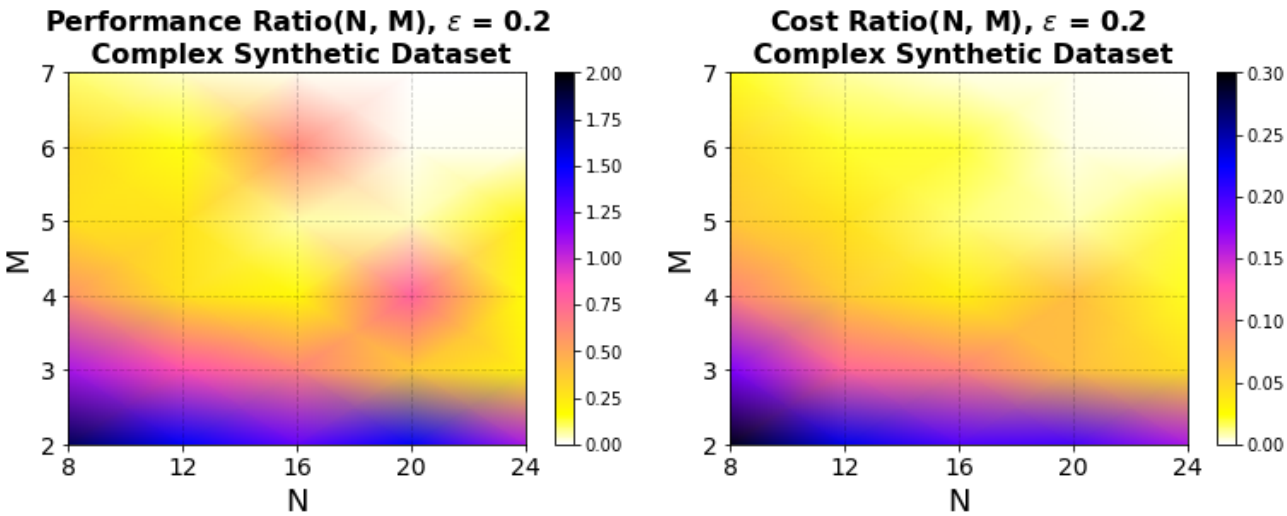


Figure 3.8: Effect of varying N and M on the performance ratio (left) and cost ratio (right) for fixed $\varepsilon = 0.2$, using the generation of seeded complex datasets to ensure sufficient complexity in the optimal molecule.

It can be seen that the indicator ratios both decreased rapidly with increasing M however the decrease with increasing N is much slower than in the simple dataset case. This indicates that the Q-learning algorithm is more effective for larger molecule problems, i.e. MDPs with a higher number of steps per episode resulting in more information to learn from. Having a large number of ligands to choose from remains desirable, however the size of the final molecule in the MDP is significantly more important in terms of minimising the resulting indicator ratios of the Q-learning algorithm.

It is observed that there are two particular datapoints that appear to be outliers, with a higher performance ratio (and slightly higher cost ratio) relative to its neighbours. Specifically, this occurs at $N = 16$, $M = 6$ and $N = 20$, $M = 4$. This could be explained by each N, M pair generating a unique seeded dataset with the possibility of some datasets being particularly challenging to solve. For example, it is possible to have some molecules in the set that have a high value similar to that of the optimal molecule but share few or none of the same ligands present in the optimal molecule. In this case the exploitative actions may be directed towards assembling these high value non-optimal molecules and the optimal molecule can only be found by a 'lucky' period of frequent exploration, resulting in higher indicator ratio values.

3.3 Physical Data

The Q-learning algorithm was then applied to two physical datasets of real ligands, molecules, and corresponding anisotropy values. The size of these datasets are limited due to the extremely high computation time required for each D value datapoint. The two physical datasets used were of the form $N = 13$, $M = 4$ for dataset 1 and $N = 206$, $M = 2$ for dataset 2 and were acquired from colleagues in the research group.

One point to note is that some of the possible molecules from the combination of ligands available are absent from the datasets. In this case the $-D$ values of the missing molecules are simply set equivalent to the worst molecule value in the set.

In particular for dataset 1 no molecules of the form $[A, A, A, B]$ or $[A, A, B, C]$ (in any order) were present[1]. The set of possible states $\mathcal{S}$ and set of possible actions $\mathcal{A}$ in the MDP were restricted accordingly such that $n_{\text{poss}} = 806$.

For dataset 2 with $N = 206$ and $M = 2$ it can be shown that $n_{\text{poss}} = 21,321$, however there were only $14,564$ present in the dataset. This means that 6,757 or approximately 31.7% of the possible molecules were absent and set to the worst possible value in the dataset. This is expected to have a negative impact on the results of the indicator ratios.

The result of varying the exploration parameter ε was investigated for dataset 1:

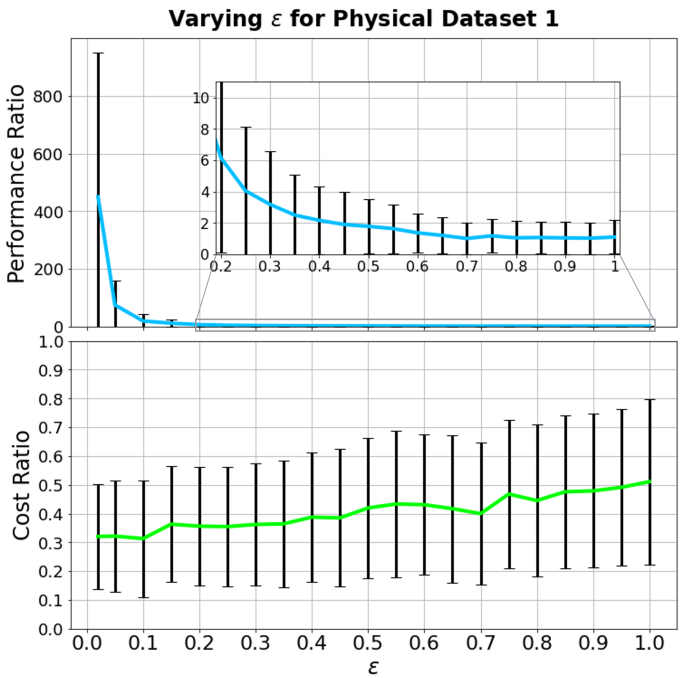


Figure 3.9: Indicator ratios vs ε for physical dataset 1.

It can be seen in regard to the performance ratio, that exploitative actions provide no improvement in the convergence rate of the Q-learning algorithm. An equal balance of exploration and exploitation has the equivalent convergence efficiency to completely random action selection. It can then be seen that as $\varepsilon \to 0$ the performance ratio diverges much earlier than in the complex synthetic dataset case (Figure 3.7). This demonstrates the increased complexity of the physical anisotropy. For example, it is common that replacing a single ligand in a high-value molecule drastically reduces its value. There are few, very particular, combinations of ligands that result in highly negative anisotropy. This implies very low correlation in anisotropy between similar molecules and it is clear that problems of this nature prove difficult for the Q-learning algorithm.

In regard to the cost ratio, it can be seen that the minimum occurs for small ε and there is a general gradual decrease in the cost ratio for decreasing ε. The variance is also high for all ε values. This indicates a high sensitivity to the stochastic nature of exploratory action selection. The minimum cost ratio attained was 0.3130 ± 0.2026 such that the construction of $31.30 \pm 20.26\%$ of the possible molecules

[1]This is due to the structure of these particular molecules being very unlikely to be synthesised in the laboratory.

was required before finding the molecule with most negative anisotropy. This is not the most impressive result, particularly when the high variance is taken into account, however, it can still be stated with some confidence that the algorithm has learned something of value to reduce the cost ratio to this minimum. It can also be extrapolated that smaller ε values would result in small improvements to the cost ratio.

It should be stressed that the size of the dataset is very small with $N = 13$ and $M = 4$. From previous analysis is it clear that a larger N, and particularly a larger M dataset would be a more appropriate application of the Q-learning algorithm and would show a significant improvement in results.

The same analysis was performed for physical dataset 2 ($N = 206$, $M = 2$):

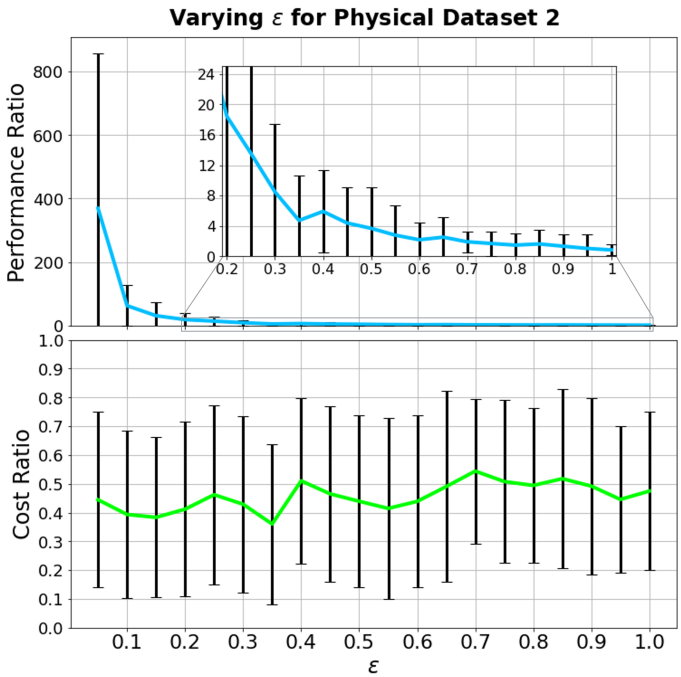


Figure 3.10: Indicator ratios vs ε for physical dataset 2.

It can again be seen in regard to the performance ratio, that exploitative actions provide no benefit in terms of the rate of convergence. The minimum value of the performance ratio, or highest convergence rate occurs for $\varepsilon = 1$, completely random action selection. This would indicate that the Q-learning algorithm has been pushed too far and cannot effectively solve this problem.

This is reinforced by observing the results for the cost ratio. A definitive improvement could not be observed within the variance for decreasing epsilon. For this particular dataset, randomly sampling the molecules is equivalently as effective as the Q-learning approach. The Q-learning algorithm has failed in this case to reduce the number of anisotropy computations required to find the optimal molecule.

This can be explained, firstly, due to the missing data, with 31.7% of possible molecules absent from the dataset and set to the worst possible value. This means that no information is gained in an episode that constructs one of these molecules. The dataset is also very small and the MDP is only 2 steps for $M = 2$. It has been shown that a high value of M is the most important contributor to minimising the indicator ratios. It can be concluded that the Q-learning algorithm failed to learn anything of value for this particular dataset.

Despite the underwhelming results for the physical datasets, this research provides strong evidence that for an appropriately sized dataset with larger N, and crucially larger M, the Q-learning algorithm has the potential to be an extremely powerful tool in vastly reducing the number of anisotropy computations required to find the most promising high performance SMM candidates with maximal negative anisotropy.

4

Conclusion

The Q-learning algorithm was successfully implemented to solve the reinforcement learning problem of constructing molecules ligand by ligand to determine, at low computational cost, the most promising single molecule magnet candidates with maximal negative anisotropy. The update scheme of the Q-learning algorithm was extended due to the nature of the problem, resulting in a significant improvement in optimization rate and required number of anisotropy computations.

The Q-learning algorithm was shown to be very effective when applied to randomly generated synthetic datasets. Simple datasets were generated such that the optimal molecule was the trivial case with repeated ligands and no interaction between ligands considered. More complex datasets were generated with a model of ligand interactions considered such that the optimal molecule was non-trivial with minimal repeated ligands. The Q-learning algorithm was shown to be most effective in reducing the number of required anisotropy computations when applied to problems in which the size of the final molecule was large. These larger problems required more steps in the Markov decision process. This allowed more information to be gained in each episode of the Q-learning process, resulting in learning improvements.

When applied to physical datasets with real anisotropy data the Q-learning algorithm failed to yield a definitive reduction in the number of required anisotropy computations. However, the two physical datasets analysed were incomplete and relatively small (low N and M parameters) such that the inherent strengths of the Q-learning algorithm were not fully exploited. Evidence would suggest that the application of Q-learning on a more suitably sized physical dataset would result in a considerable reduction in the number of required anisotropy computations. This positions the Q-learning algorithm as a potentially valuable tool in the exploration of the chemical space for high performance SMM candidates, provided it is applied under suitable conditions.

Some further research to be pursued could be coupling the Q-learning algorithm to some non-linear black box model to give computationally inexpensive approximations of the anisotropy values of molecules. In the Q-learning algorithm only values relative to one another are relevant such that high precision anisotropy values are not strictly required to effectively explore the chemical space for promising SMM candidates. This implementation would allow for efficient exploration of the chemical space without the requirement of performing anisotropy calculations upon the construction of every new molecule. This is ultimately how novel high performance SMMs would be discovered.

Another potential improvement that could be explored is the implementation of deep Q-learning, in which the Q-table is replaced with an artificial neural network (ANN) that takes states as inputs and returns action-value pairs. In this way, ligands could be represented by a feature vector with more information contained, such as coordinates or SMILES[7] chemical signature, rather than a simple identifier. This more detailed information, combined with the power of ANNs, could allow the algorithm to achieve more intelligent and efficient exploration of the chemical space.

Appendix

A.1 Figures

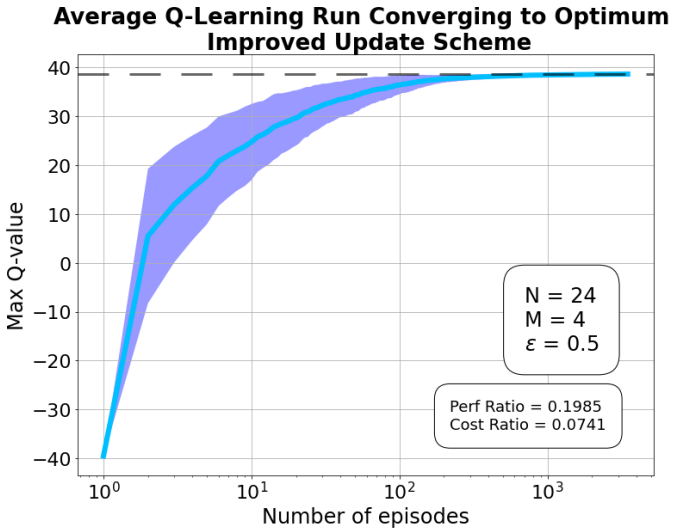


Figure A.1: Semi-log plot of an average Q-learning iteration for a simple synthetic dataset with improved Q-learning update scheme.

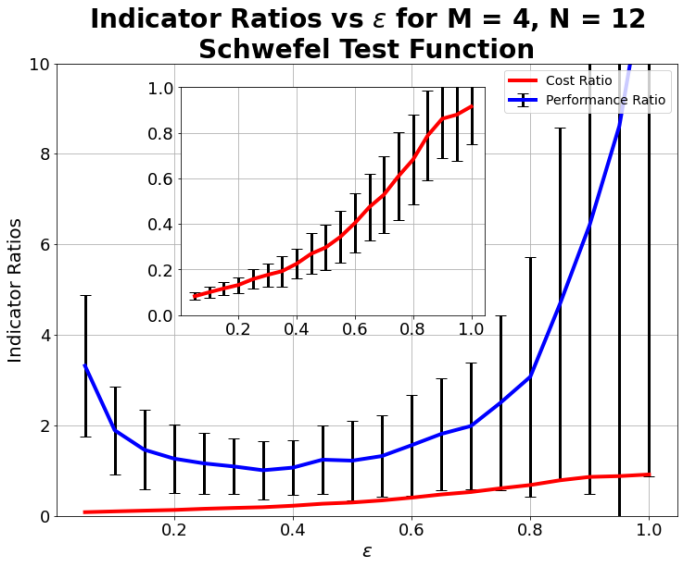


Figure A.2: Indicator ratios vs ε for synthetic dataset generated via Schwefel test function[6], prior to improved Q-learning update scheme. Similar results are observed in Figure 3.2.

A.2 Source Code

Example source code of varying ε vs indicator ratios for both simple and complex synthetic datasets is available at: `https://github.com/jim240/TCD-Final-Year-Project`.

A.3 Derivations

The rate of change of $n_{\text{poss}}(N, M) = \binom{M+N-1}{N-1} = \frac{(M+N-1)!}{(N-1)!M!}$ with respect to changes in the proximity of N and M can be investigated by computing the gradient $\vec{\boldsymbol{\nabla}} n_{\text{poss}}(N, M)$. This requires the extension of the factorial function to continuous arguments given by the gamma function $\Gamma(x) = (x-1)!$, where if x is a positive integer (as is true for this case) the derivative is given by:

$$\Gamma'(x+1) = x!\Big(-\gamma + \sum_{k=1}^{x} \frac{1}{k}\Big)$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant. This leads to the following partial derivatives and maximal rate of increase of $n_{\text{poss}}(N, M)$:

$$\frac{\partial n_{\text{poss}}(N, M)}{\partial N} = \frac{\Gamma(M+1)\Big[(M+N-1)!\,\Gamma(N)\big(-\gamma + \sum_{k=1}^{M+N-1} \frac{1}{k}\big) - (N-1)!\,\Gamma(M+N)\big(-\gamma + \sum_{k=1}^{N-1} \frac{1}{k}\big)\Big]}{\big(\Gamma(N)\,\Gamma(M+1)\big)^2}$$

$$\frac{\partial n_{\text{poss}}(N, M)}{\partial M} = \frac{\Gamma(N)\Big[(M+N-1)!\,\Gamma(M+1)\big(-\gamma + \sum_{k=1}^{M+N-1} \frac{1}{k}\big) - (M)!\,\Gamma(M+N)\big(-\gamma + \sum_{k=1}^{M} \frac{1}{k}\big)\Big]}{\big(\Gamma(N)\,\Gamma(M+1)\big)^2}$$

$$\implies \|\vec{\boldsymbol{\nabla}} n_{\text{poss}}(N, M)\| = \sqrt{\Big(\frac{\partial n_{\text{poss}}(N, M)}{\partial N}\Big)^2 + \Big(\frac{\partial n_{\text{poss}}(N, M)}{\partial M}\Big)^2}$$

with analysis showing a higher rate of increase when N and M are far removed compared to N and M in close proximity.

Bibliography

[1] N. Chilton, Lectures on molecular magnetism, Department of Chemistry, The University of Manchester, URL: https://www.nfchilton.com/teaching.html (2019).

[2] L. Thomas, F. Lionti, R. Ballou, D. Gatteschi, R. Sessoli, and B. Barbara, Nature **383**, 145 (1996).

[3] A. Zabala-Lekuona, J. M. Seco, and E. Colacio, Coordination Chemistry Reviews **441**, 213984 (2021).

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (The MIT Press, 2018).

[5] D. Silver, Lectures on reinforcement learning, URL: https://www.davidsilver.uk/teaching/ (2015).

[6] S. Surjanovic and D. Bingham, Virtual library of simulation experiments: Test functions and datasets, Retrieved February 9, 2024, from http://www.sfu.ca/~ssurjano.

[7] D. Weininger, Journal of Chemical Information and Computer Sciences **28**, 31 (1988).

[8] D. Gatteschi, R. Sessoli, and J. Villain, *Molecular Nanomagnets* (Oxford University Press, 2006).

[9] G. Rajaraman, *Computational Modelling of Molecular Nanomagnets* (Springer Cham, 2023).

[10] N. Chilton, Single-molecule magnets: Design, measurement, and theory, The University of Manchester, IES EPR Seminar Series, URL: https://www.nfchilton.com/teaching.html (2020).