Algorithmic Discovery Methods for Robust Dynamical Decoupling Sequences

George Witt*
Physics Department, University of Maryland College Park

Jner Tzern Oon and Ronald Walsworth
Physics Department, University of Maryland College Park
(Dated: April 18, 2025)

We introduce Nitrogen-Vacancy (NV) systems and our pseudo 2 level system to model them. We explain the relevance of producing high fidelity microwave control pulses in the 2 level approximation to isolating NV centers from background noise. Given the large search space for possible robust dynamical decoupling sequences for 2 level systems, we setup and apply a number of value-based, policy-based, and a model-based reinforcement learning (RL) algorithmic approaches to search for sequences. We observe how RL methods overfit to their training environment and propose a solution with a Monte-Carlo sampled environment. We present training runs for our RL algorithms for constructing 6 and 12 length sequences, comparing to literature and the full 6 length brute force distribution. We determine patterns in the 6 length pulse sequence distribution to build a beam search algorithmic search method that outperforms our best RL models. We compare our RL and beam search results against similar RL work for decoupling that produced 2 sequences: angle12, a length 12 AI-only generated sequence, and yxx24, a length 24 AI-human produced hybrid sequence [1]. We show how the patterns we find allow us to find sequences that improve in fidelity compared to length 12 pulses in literature [1] from 0.99 to approximately 0.999. Using the same method, our overall best pulse sequence achieves a fidelity of ≈ 0.99998 over 24 pulses. We propose methods through which the pulse sequence patterns determined to build our beam search technique could be further expanded for advanced RL techniques.

CONTENTS

I. Introduction 1
A. NV Systems and Simulation 1
B. Decoupling 3
C. Challenging Search Space 3

II. Methodology 4
A. Reinforcement Learning 4
B. Environments 4
1. Overfitting 5
2. Monte Carlo Sampling 6

III. Results 6
A. 6 Length Sequences 7
B. 12 Length Sequences 7
C. Pulse Sequence Patterns 9
D. Informed Brute Force 9

IV. Discussions and Conclusions 9
A. Advantages and Limitations of RL 11
B. Future Work 12

V. Acknowledgments 12

References 12

A. RL Algorithms 13
1. DQN 13
2. DQN HER 13
3. QRDQN 13
4. A2C 13
5. PPO 14
6. Dreamer V3 14

I. INTRODUCTION

Quantum sensors have broad application to tasks like magnetometry, structure characterization (strain, or Nuclear Magnetic Resonance, NMR, tasks), etc [2]. For example, the SQUID (Superconducting Quantum Interference Device) offers high sensitivities approaching values as low as 0.3 fT $\mathrm{Hz}^{-1/2}$ with mm spatial resolution, or nT sensing scales with sub-micron resolution [3, 4]. However, the cryogenic requirement of the SQUID system limits its mobility, applicability to live biological samples, high temperature/pressure environments, etc. Platforms like the Nitrogen Vacancy (NV) in diamond provide superior spatial resolution reaching resolutions of hundreds of nm over μm fields of view despite weaker magnetic sensitivity [5].

A. NV Systems and Simulation

NVs are local defects in a diamond lattice [2, 5]. The defect appears when 2 adjacent carbon atoms are removed, and one hole is filled with a nitrogen atom as shown in Figure 1A. We consider only the NV(−) de-

* Also at Computer Science Department, University of Maryland College Park

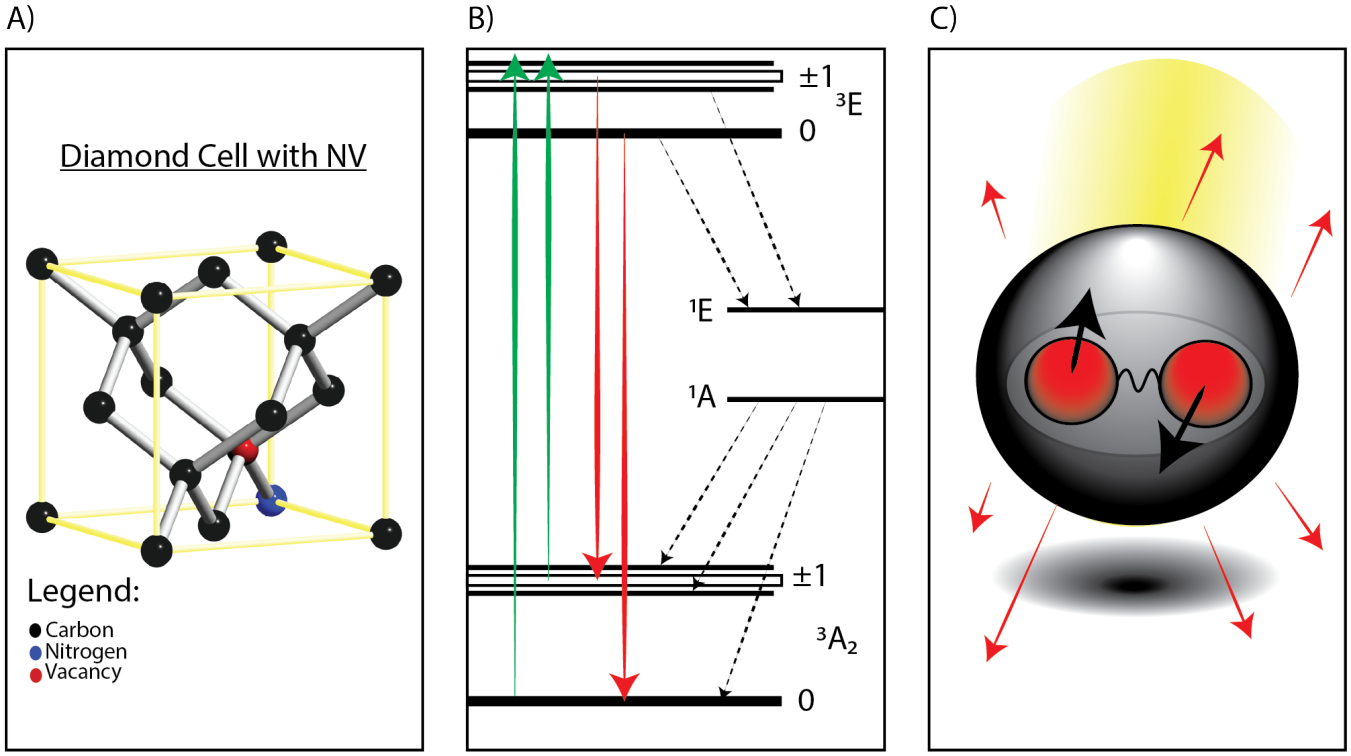


FIG. 1. (A) Represents a unit cell of diamond with a Nitrogen Vacancy - NV - present. The NV is represented as as the red replacement for the black carbon. Then one carbon is missing in the blue marked sphere. Colors are chosen here just for visualization. (B) is the energy diagram of the NV system similar to [6]. The NV system has a ground triplet state ${}^3A_2$ and an excited triplet state ${}^3E$. The transitions from the ground to excited states are usually driven by a 532 nm green laser, de-excite as red emission. Both $m_s = 0, \pm 1$ can decay through the Inter-system-crossing (ISC) states ${}^1A$ ${}^1E$ shown on the right. This de-excitation path is largely non-radiative. The transitions overall are usually spin-conserving such that driving can initialize a population in $m_s = 0, {}^3A_2$. (C) is a cartoon representing the primary sources of noise in our system. Modeled as 2 interacting spin $\frac{1}{2}$s, there is a magnetic dipolar interaction between the two spins (coupling between the 2 spins). They are also affected by local magnetic disorder in the environment magnetic field - hence the randomly oriented arrows. The applied microwave pulse is visually represented as a yellow illumination here, schematically representing that the spins can be affected by errors in the applied pulse. This cartoon summarizes visually the 3 primary sources of error.

fect [6]. This defect exhibits a rich energy level structure displayed in Figure 1B. It consists of a spin triplet state ${}^3A_2$ with spin quantum numbers $m_s = 0, m_s = \pm 1$, and a similar triplet excited state ${}^3E$ [6]. Electrons are typically excited from the ${}^3A_2$ to the ${}^3E$ state through application of a 532 nm green laser [6]. De-excitation back to ${}^3A_2$ radiates in the optical spectrum and can be experimentally read out [6]. De-excitation from the excited state ${}^3E$ can also occur through the displayed singlet states. These singlet states are shown in Figure 1B - the ${}^1A$ and ${}^1E$ states. While the optical transitions between the excited and ground triplet states are spin-conserving between the m_s states, the ${}^3E$ to singlet to ground triplet de-excitation path is usually not. An electron in the $m_s = \pm 1$ states of ${}^3E$ is more likely to de-excite through the non-optically-radiant singlet state than not, while the $m_s = 0$ states usually decay directly back to ground. Since the singlet state path, known as the intersystem crossing (ISC), flips the state from $m_s = \pm 1$ to $m_s = 0$, repeatedly pumping the electron population from ${}^3A_2$ to ${}^3E$ accumulates electrons in the $m_s = 0$ ground state. These dynamics can then be further manipulated for sensing or other tasks [2].

For example, a population of electrons prepared in the $m_s = 0, {}^3A_2$ state can be used for magnetometry. In the presence of a magnetic field, the energy levels of the $m_s = \pm 1$ states change. When this system is then driven with a microwave field on resonance with the transition from the ground $m_s = 0$ to $m_s = \pm 1$ states, the ISC mechanism leads to a reduction in observed photo-luminescence (PL). The difference in resonance frequency from the expected resonance frequency for zero-field splitting only, $\sim$ 2.87 GHz, can be used to determine the signal magnetic field. This simple example is known as ODMR [6], Optically Driven Magnetic Resonance. While there are many energy levels in the displayed diagram, the population difference in the $m_s = 0, \pm 1$ states after initialization is what affects PL - the output signal.

B. Decoupling

Decoupling is the process of isolating the NV system or sensor from background noise. Consider a background magnetic field, for example, that shifts the $m_s = \pm 1$ energy levels. Now, the frequency that was on resonance for the signal $m_s = \pm 1$ transition is slightly off resonance. Since NV centers are used in an ensemble where an entire diamond sample containing NVs is used, then small differences in these resonance shifts, phase differences between the $m_s = 0, \pm 1$ states, will wash out the PL pattern through interference. Our goal is to suppress this dephasing.

To do this, we'll use a model with 2 interacting spin $\frac{1}{2}$s like other approaches [7]. This is represented in Figure 1C. When subjected to a near-resonant microwave driving field, the NV spin transitions are limited to the selected 2 level subspace and can be modeled as a pseudo 2 level system. In our 2 level system the effect of dephasing is represented as the infidelity between the final state after time t_f and the initial state initialized at $t = 0$. So we approximate our NV system as a 2 level system with the $m_s = 0, 1$ (or -1) states, and the interactions as magnetic dipolar interactions between 2 spins.

Our goal is to determine what sequence of microwave pulses can be applied to the system to eliminate decoherence. These microwave pulses purposefully drive transitions from the $m_s = 0$ to the 1 state with the goal of returning the state to the $m_s = 0$ state as closely as possible. Viewing this in our 2 level system lens, we'll refer to our microwave pulses in terms of rotations on the Bloch sphere with the $m_s = 0, 1$ states being the pure states on the sphere. Specifically, we'll consider π and $\frac{\pi}{2}$ microwave pulses, representing rotations of π and $\frac{\pi}{2}$ radians on the Bloch sphere respectively. A larger fidelity in the final state to the initial state represents more of the population remaining in the $m_s = 0$ state after some time t, corresponding to a state that has been protected from noise. If the fidelity of the final state compared to the $m_s = 0$ direction is greater after applying the pulse sequence than with no control applied, we've successfully decoupled some or all of the noise.

We consider overall 3 sources of noise: (a) magnetic dipolar interaction between the 2 spins, (b) imperfect pulse rotations (c) variations in the background field, which we refer to as Magnetic Disorder. The chosen Hamiltonians for interaction and pulse rotations are presented in equation (1). Imperfect pulse rotations are represented as a small change in the unitary evolution of ψ as discussed around equation (2).

$$H = -2\pi J \left[S_z \otimes S_z - S_x \otimes S_x - S_y \otimes S_y\right] + 2\pi J_{dis} \left[S_z \otimes I + I \otimes S_z\right] \quad (1)$$

where J, J_{dis} represent interaction and magnetic disorder strengths respectively, and S_i represent the spin 1 spin operators along all 3 directions x, y, z.

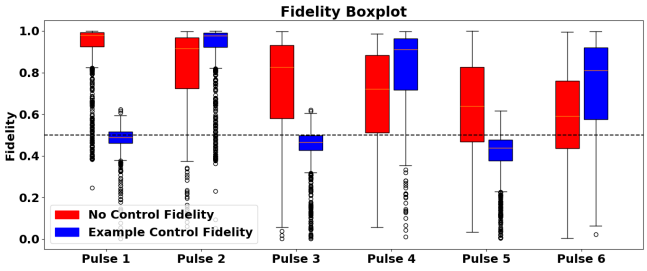


FIG. 2. We sample many 2 spin systems from physically informed distributions of possible values; the reasoning for this is discussed in section II B. The system is left to evolve under no control for 6 discrete steps of τ, represented by the 6 box plot distributions shown in red. The distribution slowly dephases from the initial coherent state. In blue, we apply a simple sequence consisting of alternating $\frac{\pi}{2}$ pulses, demonstrating how the fidelity is more maintained through this sequence. This example demonstrates how application of a simple control pulse sequence can better maintain the fidelity of our interacting systems, even under the presence of rotation errors, local magnetic disorder, and the spin interactions.

This formulation has direct application to NMR with NV systems, or even other fields like quantum computing. In quantum sensing, for example, these protocols we produce would shield the NV from background fields from the spin bath or other time-varying fields.

The need for these 'protection' pulse sequences is illustrated in Figure 2, where we compare the final fidelities of many sampled 2 level systems after applying a series of $\frac{\pi}{2}$ pulses (in blue) compared to no control whatsoever. These sampled 2 level systems each have different noise parameters for reasons discussed in section II B, hence we have a distribution of fidelities after applying each pulse. While the fidelity is lowered *during* the pulse sequence, the median fidelity is significantly higher in the controlled case against the no control distributions which slowly reduce in fidelty.

C. Challenging Search Space

The first challenge is the space of possible pulses. We've previously mentioned only π and $\frac{\pi}{2}$ pulses for this 2 level system. Of course, it is possible to theoretically rotate the state of a two level system on the Bloch sphere by any angle. But, as the search space of such arbitrary pulses is arbitrarily large, we start by limiting our work to these predefined - and experimentally feasible - π and $\frac{\pi}{2}$ pulses only.

We choose to model a pulse sequence as N pulses, each separated by a free evolution interval τ. Adding 'no pulse' as one possible choice, we have 9 choices for each N pulses. This gives a total space of possible combinations of 9^N sequences. Considering arbitrary length sequences would make this problem intractable, so we enforce this choice to evaluate our algorithmic methods.

We therefore only consider pulse sequences of fixed length and compare to other sequences of similar length. We consider lengths of 6, 12, and 24, to compare with either brute forced distributions or prior work.

This search would still be impossible with brute force or algorithmic searches only for long sequences. Limited by the simulation time on the CPU, evaluating a reasonable fraction of $9^{12}, 9^{24}$ pulses is not possible. Therefore, we will primarily consider 1 efficient search technique known as **Reinforcement Learning** - see [8]. Reinforcement Learning is discussed at length in Section II A.

II. METHODOLOGY

We mostly use RL techniques in this project. These techniques are elaborated on at length in Section II A. Our 'environment' - simulator - is explained in Section II B.

A. Reinforcement Learning

Reinforcement Learning is a subset of the field of Machine Learning (ML) [8]. There are 3 regimes in ML. The first is Supervised Learning (SL), in which a model learns to approximate a mapping function from some a set of inputs - X - to a set of known correct outputs - Y. Linear regression is an excellent example of this. The second is Unsupervised Learning (UL), where a model learns to compute a desired output Y using only the set of inputs X. Any encoding problem is an example of UL. The third is Reinforcement Learning (RL), in which a measured metric is maximized in a defined simulator [8]. This metric is known as the 'reward', and is represented in our case as

$$R = -\log\left(1 - \mathrm{Re}\left[\frac{\mathrm{Tr}(U_1 U_2^\dagger)}{\mathrm{Tr}(U_2 U_2^\dagger)}\right]\right) \tag{2}$$

where $\psi(t) = U_1(t)\psi_0$ and $\psi_0 = U_2\psi_0$, therefore $U_2 = \mathrm{I}$. U_1 represents a unitary evolution operator that evolves ψ_0 to $\psi(t)$ encapsulating the evolution due to all pulses in a sequence.

The process of RL is summarized in Figure 3, and the 3 major components, environment, agent, buffer, are explained here. The first component, the environment, is our simulator. The environment simulation must occur over discrete steps. In our simulator, this corresponds to individual discrete choices for pulses to apply. The environment must also be represented at each discrete step with a state s_t. This state vector contains all information about the environment. Our state vector is represented as the expectation and variance (for why variance, see Figure 5) of our combined Pauli operators $P_{ij} = S_i \otimes S_j$ on the quantum state $\psi(t)$.

The state vector must also follow the Markov-Chain property: it should be possible to determine s_{t+1} using only a_t, s_t. The value of each state must be evaluated using a reward function which is given in our case in equation (2). If the state vectors do not have the Markov-Chain property, there is no reasonable expectation that a model should be able to predict the next immediate best action given the current system state.

The state and reward (optionally) are passed to the agent. The agent represents the technique that is used to produce each pulse (action). Using the state, the agent create a **policy** representing a rule-set by which the next action in the sequence is chosen. This action is chosen, sent back to the environment for simulation, and the process is repeated. The process ends only after each pulse in the sequence is chosen sequentially.

The records of all s_t, a_t, R_t are passed to the buffer. This list of actions, rewards, and states is known as a *trajectory*. Trajectories are sampled from the buffer in a way defined by the agent, run through the simulator if necessary, and then used to update the parameters of the agent as defined by the agent method. Almost all update rules for the RL models described in the appendix are for trajectories sampled from the buffer.

Agents can be designed in two primary ways, 'model-free' or 'model-based' (Figures 3B, C, respectively). Model-Free RL models use some combination of neural networks to produce the policy using only the past real observed state. In a simple example, the state comes into a neural network, is processed, and then produces a probability distribution over possible actions. The final choice of action is sampled from this policy probability distribution. Model-based RL models are different. They take in a state, process it, then use their own internal network to simulate applying a series of further actions (pulses). They predict the states for each pulse in a sequence using their own *model*, *World* model, of the environment dynamics. This world model is learned along with the RL policy during training. While we evaluate a number of agents, we leave the mathematical details of each agent to the appendix.

We employ 2 broad classes of model-free agents along with 1 model-based agent. The model-free classes are 'policy networks' and 'value networks'. Policy networks produce a probability distribution over possible actions at each step and sample the next action from the distribution. Value agents produce an estimate of the 'value' of each action at every step and then use a greedy policy to select the best-valued action. Our 1 model-based agent follows the Dreamer framework (see appendix) unlike well-known techniques such as the AlphaZero/Go series of models [9, 10].

B. Environments

Our 2 level model is initialized in the ground state direction ($m_s = 0$ initialization) at the start of each se-

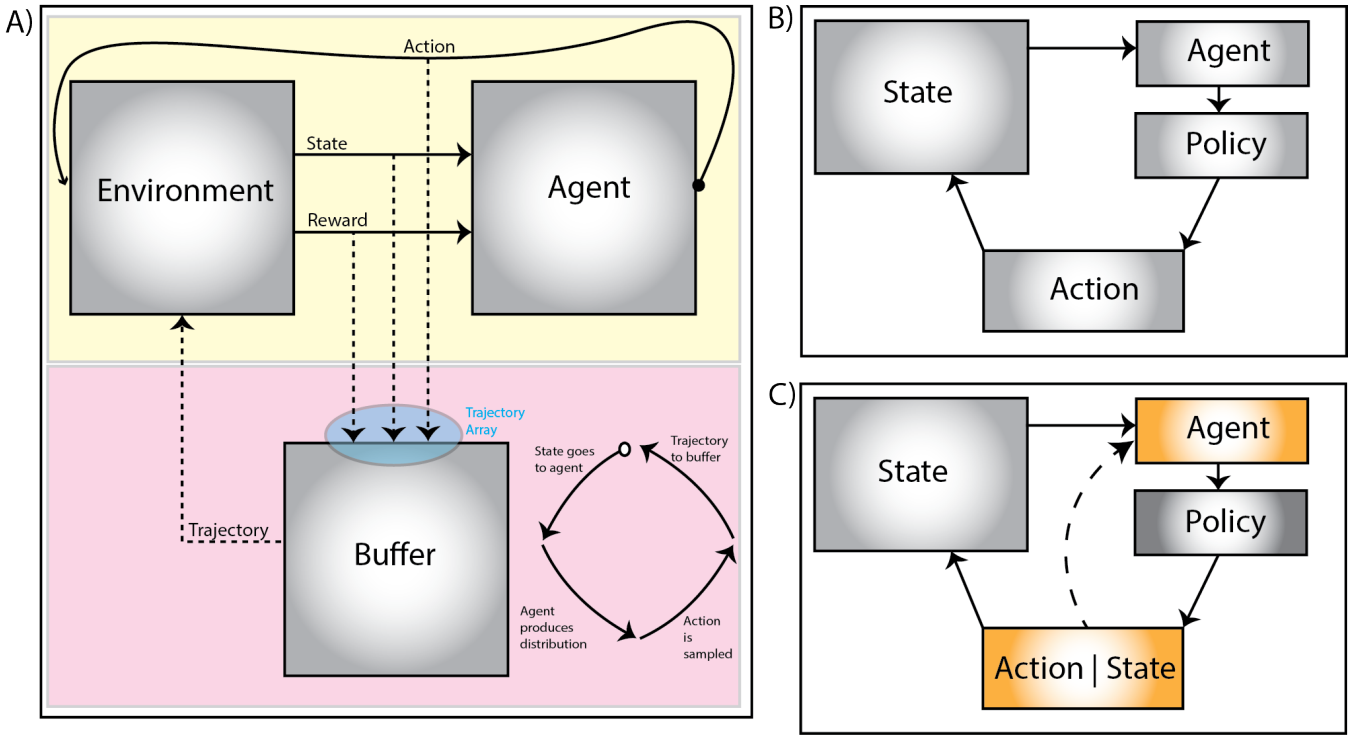


FIG. 3. This figure schematically explains the reinforcement learning (RL) process. The overall method is showcased in (A) between the 3 major components: the Environment, Agent, and Buffer. The environment - our simulator - is described by a state s_t with some value, known as the reward, at all times. This is supplied to the agent which produces an action and the discretized pulse sequence construction process proceeds. These values are all stored in the buffer in an order known as a 'trajectory'. (B) This describes the most common agent design, 'model-free'. In a model-free agent, the state is passed to some general agent definition (usually 1 or 2 neural networks) denoted again as 'agent'. This agent then produces some output that is used by the policy to create an action. (C) In model-based RL, shown here, the state is processed by the agent, through the policy, to create both a predicted action and predicted next state. This predicted next state is fed back into the agent to predict again, until the end of the trajectory. These predictions are then used in some defined way to determine the *real* action to apply to the state. The parts that change from model-free RL have been highlighted in orange.

quence. The simulation is done by determining (a) the constant initial state, (b) the unitary operators corresponding to all possible pulses and free evolution time, (c) the integer list of pulses to test, (d) computing the unitary operator corresponding to the entire pulse sequence, (e) applying the operator to evolve the state and computing a fidelity reward with equation (2).

The values for the interaction strength, angle error, disorder, are all selected a-priori. We choose values similar to the NV system, but our program is generalizable to any other noise parameters desired.

1. *Overfitting*

We applied Deep Q Networks (DQN) and the DQN HER modification to create length 24 sequences and compare with our reference 24 length sequence yxx24 [1]. Note again that the two sequences we compare against, yxx24 and angle12, were produced either as a mix of human and AI development (yxx24) or through AI training only (angle12) [1]. We choose these sequences to compare against due to the project's similarity to our work.

Our training was performed on an environment with a Rabi frequency of 20 MHz, an interaction strength of 0.1 MHz, disorder of 0.01 MHz, and a 5% angle error applied to each pulse, chosen based on results in [11–13]. We found a sequence using only RL that outperformed our reference sequence yxx24 [1]. However, Figure 4 shows our created sequence does not generalize well to systems outside of the model training environment.

In Figure 4, our generated sequence in blue outperforms the reference sequence in an environment with 5% angle error. However, it underperforms in environments with any other setting. This same pattern is present for the other noise settings. Our sequence was best within the training environment, but the created sequence is not truly robust to general noise patterns. In a real bath where the noise parameters may vary greatly between NVs due to separation, interaction with the whole sample, crystallographic orientations, etc. this sequence creation method would not result in a *robust* sequence.

To encourage our models to learn robust sequences, we introduce the idea of Monte Carlo Sampling our testing

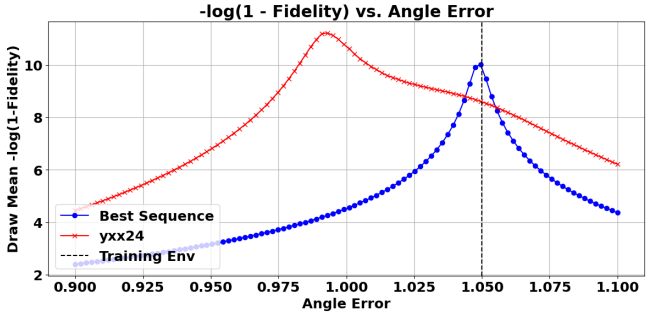


FIG. 4. Here we plot our produced sequence in blue and the reference sequence yxx24 in red. Our training environment is displayed with a vertical dashed line. Observe our model's sequence outperforms yxx24 in the trained environment with 5% angle error, however it does not generalize well to other environments. This is overfitting to the training environment.

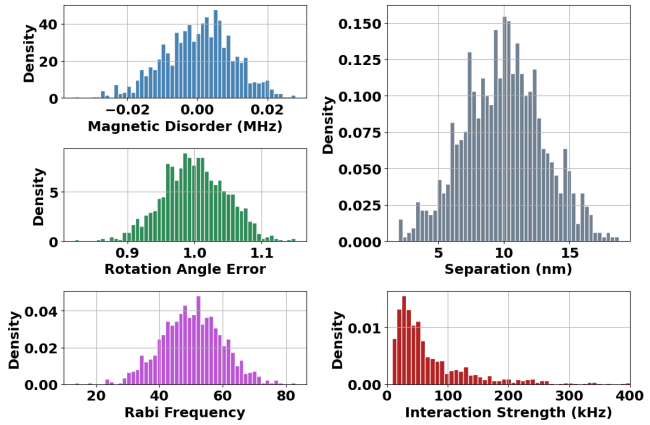


FIG. 5. We present an example of our sampled environments with 1000 draws. Observe we sample the disorder, rotation angle error, Rabi Frequency, and separation from Gaussian distributions. The interaction strength is computed from the sampled separation using equation (3).

environment.

2. *Monte Carlo Sampling*

In our Monte Carlo environment - MC environment - we use many **draws** of our 2 level simulator at once. Each individual draw is identical in form to what was used before with noise settings individually sampled from predefined population distributions. This produces a distribution of rewards - fidelities - for each evaluated pulse sequence. The average reward is used as the reward signal for our RL models.

An example of such a sampled MC environment with 1000 total draws is presented in Figure 5.

The disorder, angle error, Rabi frequency, and separation values are sampled from Gaussian distributions. The means and the standard deviations of the distributions are reported in Table I. For the rest of our discussion, we use the same sampled environment listed in Table I unless explicitly noted otherwise.

TABLE I. We summarize the population distribution settings for the Gaussian distributions we sample. The mean is displayed in the center column and the standard deviation in the right column. These values are chosen from results in [11–13].

Parameter	Mean	Standard Deviation
Disorder (MHz)	0	0.01
Separation (nm)	10	3
Rabi Frequency (MHz)	50	10
Pulse Error (%)	0	5

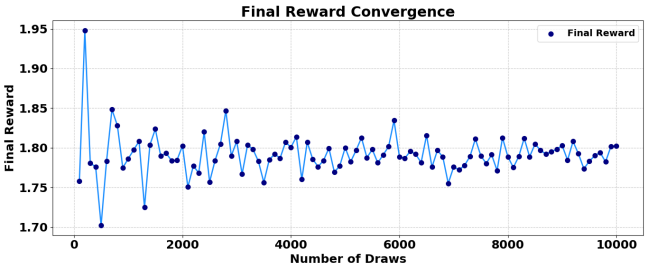


FIG. 6. We schematically observe convergence in evaluating our environment on arbitrary sequences (arbitrary sequence chosen randomly here) by increasing the number of draws for our environment and observing the computed reward approaching a constant.

Note that the interaction strength in Figure 5 can be computed from the separation with equation (3)

$$J = \frac{J_0 q}{4\pi x^3} \tag{3}$$

where $J_0 = \mu_0 \mu_e^2 \frac{\hbar}{4\pi}$ [14]. Here q is negative for the aligned NVs, μ_e is the gyromagnetic ratio for electrons, and x is the separation [14].

We limit our work to the 1000 draws mentioned as we consider it a reasonable cutoff to balance generalization and performance. Figure 6 shows how the reward calculation convergences for an increasing draw count. The RL model training requires evaluating thousands ($\sim 10^4, 10^5$) of sequences which is limited by calculation speed with higher draw counts.

III. RESULTS

Again, we present the training of 5 models in total. We choose 2 value-based model-free agents, 2 policy-based model-free agents, and 1 large scale model-based agent. These value-based agents are: DQN (DQN) and Quartile DQN (QRDQN) [15, 16]. The policy-based agents are: A2C (Actor to Critic) and PPO (Proximal Policy Optimization) [17, 18]. The model-based agent is the DreamerV3 architecture [10]. Other model variants showed similar or worse results to the ones presented here.

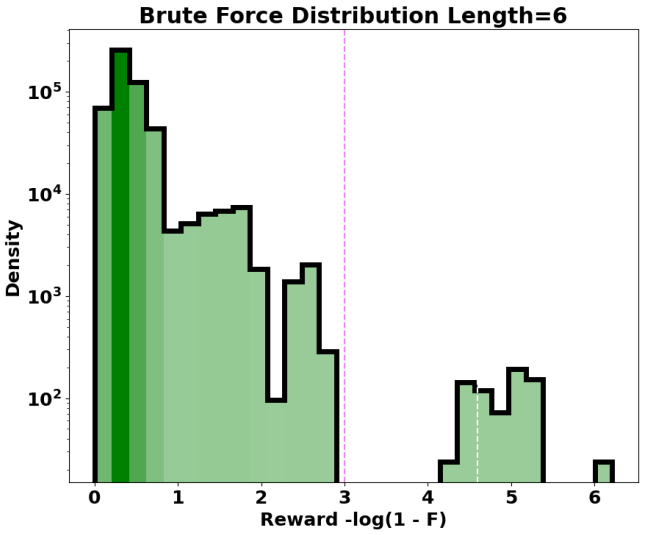


FIG. 7. We display in histogram form the distribution of all pulse sequences of length 6. The Reward -log(1 - F), with F as the fidelity, is shown on the x axis, and the relative density on the y. As expected, there are orders of magnitude more sequences with very low fidelity than those with high fidelity. We observe a discontinuity in the space with a large gap between a fidelity of 0.95 and 0.98 marked with the pink line at 0.95 fidelity. In this jump, the relative rarity of the sequences increases by an order of magnitude. Producing sequences to the right of the dividing line is a relative win considering the rarity of those sequences.

The value-based DQN approach was the original standard in the RL field and is largely considered the first successful application of *deep* reinforcement learning, RL with neural networks. QRDQN is presented as an improved variant of DQN that better balances exploitation and exploration by predicting a distribution of values, not scalar values, for each action. Policy-based networks are widely considered an improvement on value networks for their ability to handle exploration intrinsically in modeling a probability distribution. We employ A2C, one of the first policy-based networks, and PPO, a standard strong model in the RL field.

To understand our model performance, we need a small tractable problem. We brute force the 6 length pulse sequence distribution - our 'toy' scale problem - and present our results in Figure 7. We then scale our training up to 12 length pulse sequences. Greater than 24 length results are not presented as the relative scale of the number of pulse sequences that can be evaluated during a training run is too small.

All training takes place in the *same* sampled 1000 draw environment to minimize variance in training. For the 6 length case we present 15 training runs for each model unless otherwise specified. In the 12 length case, we present 3 training runs for each model.

TABLE II. We display a few conversions between reward $-\log(1-F)$ and the Fidelity F. This table is meant to be an in-text reference for reading the results figures. This is because differentiating between fidelities of 0.98, 0.993, 0.997, 0.999, etc. is all difficult if not presented on a log scale.

Reward (-log(1-F))	Approx. Fidelity
1	0.63
2	0.86
3	0.95
4	0.98
5	0.993
6	0.997

A. 6 Length Sequences

The design of the reward function is crucial to strong model performance. The most naive design is to simply use equation (2), yet this does not handle the sudden decrease in fidelity observed after many necessary pulses. To understand the best reward function design, and the performance of our models, we present the best model sequences produced over time in training in Figure 8. We present results for both the reward function in equation (2) (non-sparse), and the same reward function with *no* intermediate reward (sparse); the reward provided to the model is 0 except after the sequence is complete. All values displayed show the reward of the final produced sequence - each point corresponds to a single sequence produced.

Refer to Table II to quickly convert from our log scale fidelity to the final fidelity value.

In Figure 8 we observe a difference in training between the two reward functions. The non-sparse variant of equation (2), shows no improvement past an average fidelity of 0.63. In the sparse reward case, PPO successfully finds the best pulse sequences in the space, based on Figure 7. PPO finds these sequences after only searching about 1.5% of the total number of possible sequences (9^6). This is surprising as usually expect RL to require a smooth reward function to be feasible. Despite this restraint, PPO is still able to find optimal sequences in the small cluster of optimal sequences (roughly $1/100,000$).

B. 12 Length Sequences

To understand if this scales, we repeat the same experiment on the 12 length problem using the sparse reward. All results are summarized in Figure 9. Once again, we observe that Dreamer and PPO are the only successful models. PPO and Dreamer both achieve sequences that achieve approximately a $0.99 - 0.993$ fidelity. Dreamer's learning is again faster, with respect to the number of produced sequences, and more stable. PPO is still prone to 'catastrophic forgetting', where it has to redo progress in learning after forgetting how to produce the rare high

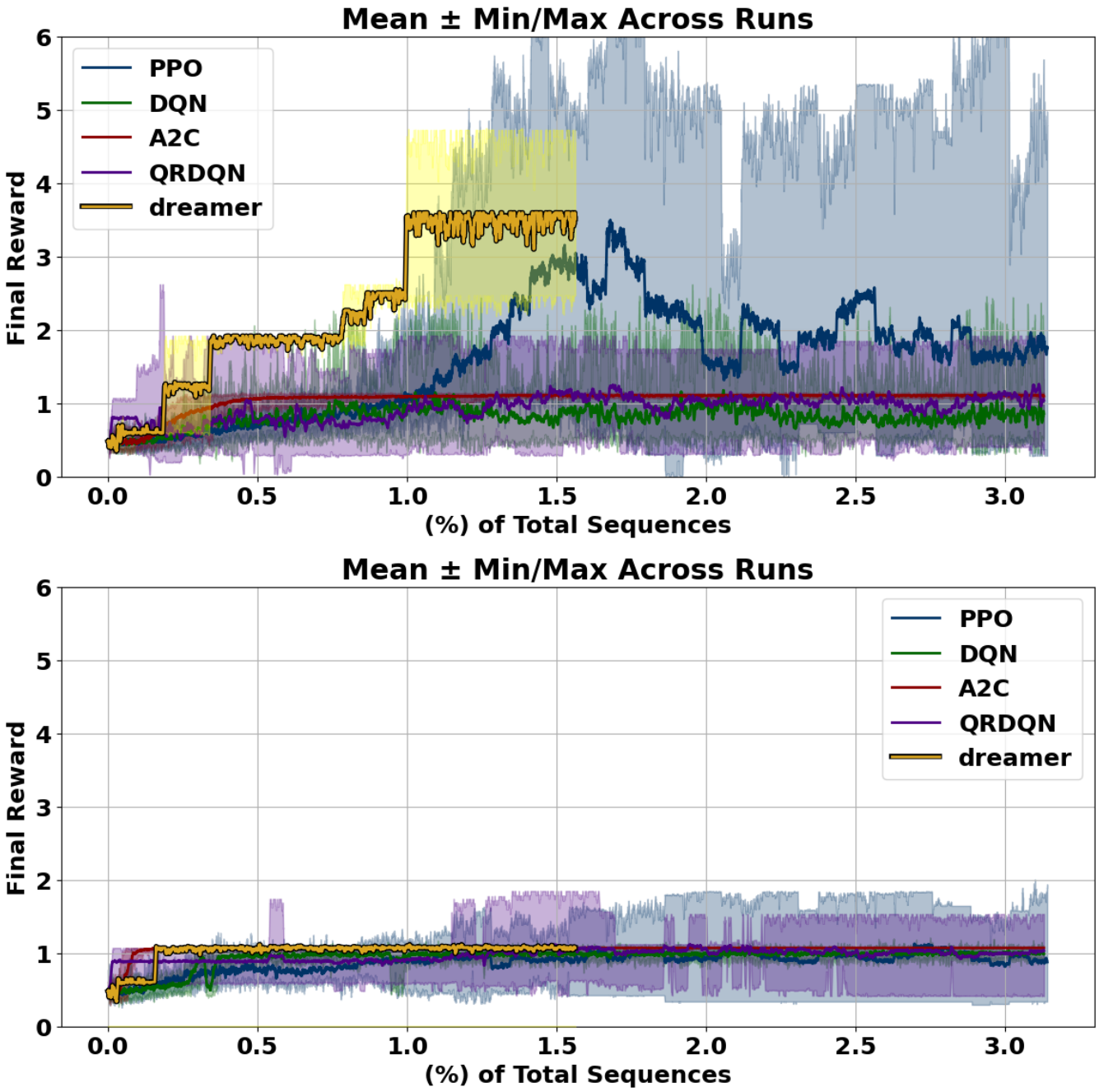


FIG. 8. This figure summarizes our results for all models in the 6 length case with a sparse reward (top) and non-sparse reward (bottom). In both plots, we show the reward of each sequence generated during training as a function of how many sequences in the space we've processed (high bound). We show the average of 15 runs for each model (3 for Dreamer due to computational complexity) as the marked line and shade between the maximum and minimum reward produced at that timestep for all other training models. In training, the only difference between the top and bottom plot is that the reward function in the bottom provides $-\log(1-F)$ at every timestep of training, vs only the end in the top plot.

reward sequences.

The best produced sequence is compared to the angle12 sequence in Figure 10. We plot the reward distribution for 100,000 new draws for the two sequences. We observe a marginally higher average fidelity for our sequence. The tail on our distribution is smaller than the angle12 tail. This reduces the relative difference in the average fidelity of our pulse sequence distribution and that of angle12 [1].

While overall conclusions will be discussed in Section IV, the problem is becoming exponentially harder with increasing pulse length. We showed in Figure 8 that it

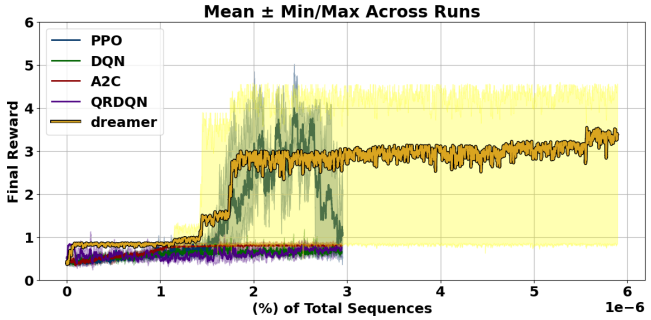


FIG. 9. Scaling to the 12 length case, we show the exact same plot as in Figure 8. We run only 3 copies of each model due to the computational complexity of the space. Dreamer was left to run for an increased number of steps to test its reliance on step count. We shade from the minimum to maximum reward achieved at each timestep across all models, and plot against a high bound estimate of the percent of all total sequences processed (9^{12})

is possible for RL to determine the optimal sequences in a space using a small percentage of the total possible sequences, but the percentage of total sequences processed is so small in our 12 and 24 length sequence cases (not shown) that results become progressively worse. We have achieved sequences with 0.99 or 0.997 fidelity in the 24 length case, though this takes significantly more time and is more sensitive to agent settings (hyperparameters) than in the 12 length case. We need a method that scales in a better way with respect to the pulse sequence length.

C. Pulse Sequence Patterns

Hoping to gain insightful lessons from the best sequences in the 6 length problem, we investigated patterns in the best pulse sequences. We applied 300 iterations of K-Means clustering to produce 4 unique clusters of reward for all 6 length pulse sequences. Information on K-Means clustering is provided in [19], though very simply it collects groups (clusters) in the data that have similar rewards. These clusters we produced are visualized in Figure 11. One cluster has a significantly higher and tight distribution of rewards. This cluster consists of the highest performing (and rare) sequences in Figure 7.

Motivated by the existence of a stand-alone separable cluster that contains all the best sequences in the space, we investigated the pulse sequence structure of the cluster in Figure 12. Using parallel coordinates, we plotted all sequences in cluster 3, colored by the maximum performance of a sequence that contains those actions. We observed a clear trend in sequence structure of alternating pulse direction $\frac{\pi}{2}$ pulses with a following π pulse. Given prevalent structure similarities in the pulse sequences, we determined a metric to measure the similarity within each cluster.

This metric is a 'Hamming distance' equivalent. We count the number of changed pulses between sequences as the 'Hamming' distance in this space; we treat each pulse as a bit and count different pulses between all unique pairs P_i, P_j in each cluster. We plot these values in Figure 13, showing the relative counting frequency of the different possible Hamming distances in each cluster. Immediately, it is clear that a significant number of pulse sequences in cluster 3 - the highest performing cluster in Figure 11 - are very similar to one another unlike in the other clusters.

D. Informed Brute Force

The existence of these patterns motivated us to improve our search methods. So far, we use no intrinsic knowledge of the pulse sequence problem to inform our model training. We have been relying on the *RL* methods themselves to determine optimal sequences. Discussion in sections III A III B showed however that these methods scale poorly with the length of the pulse sequence as they require, worst case, exponential increases in training time. So, we begin by testing an informed brute force search - beam search - on the 12 length design problem.

Beam search is a modified version of brute force. We start with a random or specified sequence. We then determine all sequences within k modifications; k is fixed to 4 for the 12 length problem. Next, we evaluate all sequences in our environment. This group of all sequences is defined as a *beam*. We then evaluate n_b of the best sequences found, and add the best n_b sequences from the beam into a list of sequences to *not* search. This taboo list prevents the beam search from looping on itself; we will not evaluate sequences on the taboo list. We do this recursively, taking each of the top n_b sequences as starter sequences for more of these *beam* searches. This effectively searches the best nearest neighbors of pulse sequences within a distance k. However, this process is limited by its inability to change more than k pulses at once and therefore cannot leap to a more creative sequence when stuck.

Our results are best encapsulated in Figure 10. The sequence displayed in Figure 10 we find after running our beam search with $k = 4$, $n_b = 10$ for only 10 iterations on the 12 length problem. As a high-end bound on the number of sequences evaluated, this means we only evaluate 0.000125% of all possible sequences - only double that of the RL in Figure 9 - but achieve superior results. Using our same beam search technique to *quickly* (< 100 iterations) improve on our reference sequence, yxx24 [1], we achieve a 0.99998 fidelity vs the reference score of 0.99997.

IV. DISCUSSIONS AND CONCLUSIONS

In section III, we presented results for a number of RL models including value-based, policy-based, and model-

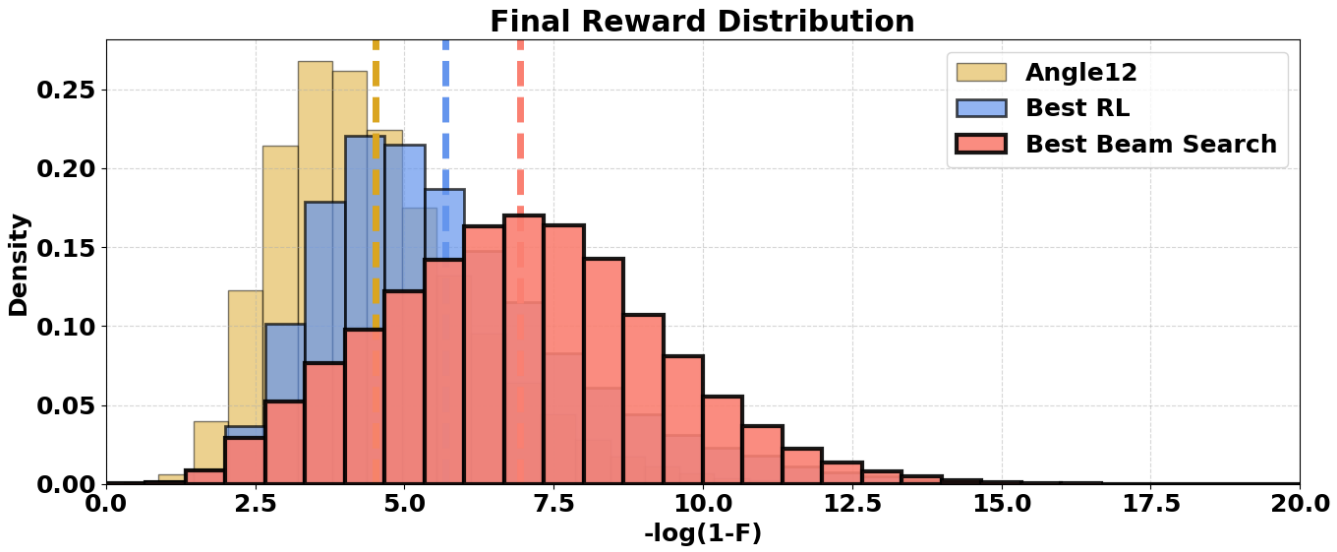


FIG. 10. Our produced best sequence in the 12 length case, from the PPO model, is compared with our best produced sequence using beam search (see section III D) and the best RL sequence we know of from literature - angle12. We evaluate all sequences in the same MC environment with 100,000 draws; this is greater than the 1000 used for training. We plot a histogram of the reward density distribution for the 100,000 draws for all sequences. We observe a significantly calmer and smoother distribution for our best beam search sequence, though both our RL and beam search method outperform previous literature approaches.

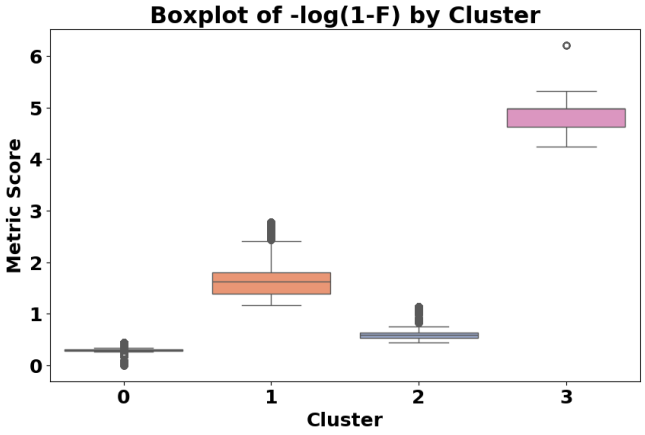


FIG. 11. Using our brute force distribution for all 6 length pulse sequences, we use a K-Means algorithm to cluster the results by reward. These clusters are drawn as boxplots here, with the arbitrary integer cluster index on the x axis and the reward - metric - distribution on the y-axis. Most sequences in the best cluster are scores around 5, similar to what PPO finds after processing roughly 1% of all total sequences in Figure 8. There is 1 standout that achieves a 6 $\sim$ 0.997 fidelity. Note that the size of each cluster is *not* enforced to be the same.

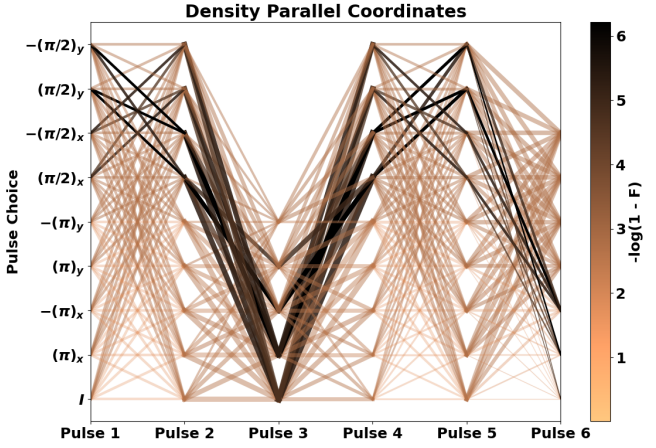


FIG. 12. The pulse sequences in the best cluster, cluster 3, are shown here. In our parallel coordinates form, each pulse sequence is formed by line segments according to the pulse index on the x axis, and the pulse choice on the y axis. I.e. the sequence $\pi_x, \pi_x, -\pi/2_y$... would have a line segment from π_x to π_x from Pulse 1 to Pulse 2, then one from π_x to $\pi/2_y$ from Pulse 2 to Pulse 3, etc. The size of the line is determined by the relative density of sequences that *share* that line segment. The color is determined by the maximum reward of all sequences that share that line segment. A clear trend from $\pi/2$ to π switching is observed and discussed.

based approaches. We observed significant differences between the performances of these approaches. Value-based models, the original deep RL technique, do not appear to *exploit* the space well-enough to produce valuable sequences. The issue is that minor changes in the best pulse sequence currently supported by the trained policy lead to significant differences in reward. These significant reward differences lead to large model gradients and unstable training. We observe the same results for the QRDQN. While QRDQN is able to learn in a more stable fashion than DQN, it is still not able to exploit the sequences it finds with a reward of 2 to pro-

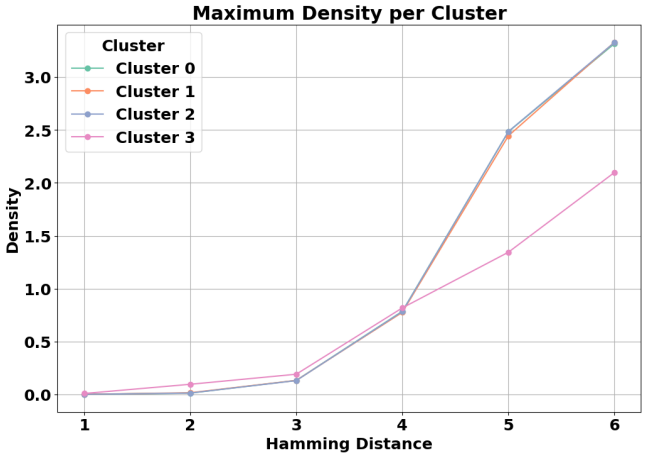


FIG. 13. We define our pulse sequence Hamming distance as the number of pulses that change between two pulse sequences. We compute this Hamming distance between all unique pulse sequence pairs in each cluster from Figure 11. We compute the density for how often each Hamming distance value occurs (maximum of 6 for 6 length sequences) and plot them as a connected line plot. We observe significantly more of the best cluster are related to one-another vs in the other clusters.

duce higher fidelity sequences. In general, we observe our value-networks to be exploring the space well, but they are fundamentally too instable with the sparse rewards to learn efficiently.

We also observed differences in training the policy-based approaches. PPO performed the best of all models and A2C performed consistently poorly. PPO's primary difference from A2C is that the model gradient is clipped to a maximum value. This clipping appears to lead to better performance as poorly performing sequences produced immediately after strong ones do not greatly affect the overall agent policy. However, this clipping does not scale well with the availability of rewards present in the space (displayed in Figure 7). Observe that as the best rewards begin hitting values above 3, which corresponds to a 0.95 fidelity, the training trajectories become noisier in Figure 8. This is partly due to the availability of possible high reward sequences falling significantly. Observe the pink cutoff line in Figure 7, where all sequences with higher than 0.95 fidelity are up to 4 orders of magnitude less likely to be found than sequences with < 0.95 fidelity. At this point, any small changes in the policy or random samples from the policy will more likely than not produce very poorly performing sequences. Since the number of updates with a negative influence (updates due to low reward trajectories) increases significantly as the maximum trajectory reward gets higher, and there are cases of repeating the same ideal sequence (overfitting to one sequence), repeated large gradients may drive the model to catastrophically forget its best sequences. This may be the same root issue that impacts A2C's training, but is not significant until the maximum reward of the model gets large for PPO. We did not investigate if decreasing the clipping threshold as a function of maximum reward would improve PPO model performance, though it would be an interesting experiment for future work.

The model-based technique is more stable than the policy-based techniques. Dreamer's sequence choice is based on its own prediction of what effect each pulse will have through the sequence. Thus, as the world model converges, the Dreamer agent is able to predict the existence of better and better pulse sequences. These sequences appear to act as checkpoints for the Dreamer model which stays stuck on the same sequence for many update steps. This more stable evolution does not appear to suffer at all from any form of forgetting, and is consistent across multiple repeated model runs. However, there is no indicator *when* the Dreamer model will find one of these checkpoint sequences for a given problem, indicating that it may need to be run for an undetermined amount of time before finding the optimal solution. The original Dreamer work, for example, needed to run over 100 million steps before achieving success on their hardest problems; this is a computational scale that is not achievable with the hardware used for this work. We are regularly running for only 50,000 to 200,000 total steps. The results for Dreamer in Figures 8 9 are for 50,000 update steps only.

A. Advantages and Limitations of RL

Overall, our results in Figures 8 9 with RL show that models can determine optimal or strong sequences over a small number of training steps. However, Figure 4 also demonstrated how sensitive RL methods are to the training environment, and Figures 8 9 to procedural choices like reward function. Any environment with noise distributions that do not resemble the population environments will fail in the more generalized tests presented in Figure 10. Our RL models are capable of determining the best sequences in a small space efficiently, even with a sparse reward, but cannot realistically process enough of larger pulse sequences to produce optimal results *without* careful fine-tuning of model parameters. Smaller qualitative experiments with time between model updates (as another example) showed that models are significantly more prone to being stuck in low performing local minima (low fidelity sequences) based on exploration settings.

Therefore, the RL techniques we showcase are highly sample-efficient yet very dependent on both the environmental noise parameters and hyperparameters for training. This can include minor decreases or increases in performance to whether or not the problem is even solvable.

B. Future Work

However, there are many avenues of work we've found that show potential to improve RL model performance. Already our limited early beam search results with a small search space show that it is possible to find sequences that significantly outperform the best RL-designed long sequences simply by naively utilizing the similarities in strong sequences. Performing the full brute force beam search is, however, time-consuming. Since RL models can efficiently search a small space with a sparse reward, either utilizing composite pulse design to make designing long sequences a smaller space or performing open-loop modification could further improve RL performance.

In a composite design problem, fundamental patterns determined in the best sequences shown in Figure 12 would be used to limit the number of total choices. For example, building a sequence that is of length 12 requires picking 12 actions with no feedback on their performance. If, instead, composite choices are provided to the model where the model picks between *3* (ex) at a time pulses, like a $\frac{\pi}{2}, \frac{\pi}{2}, \pi$ structure shown in Figure 12, then designing a 12 length sequence is similar to designing a length 4 sequence. These composite pulses would need to be designed based on considerations in modern literature, patterns in current control sequences, and results of small beam searches. As an example flow, imagine constructing the best 24 length sequence in blocks of 4 (6 pulses each). In 2 of these blocks, the best 6 length sequence could be used and the RL fill in the other 12 pulses. This approach is one example of a potential architecture that could limit the design space size to something manageable for current RL approaches.

One of the fundamental issues with the RL technique we presented in this work is the 'credit assignment' problem. Since the reward is 0 at every step of pulse sequence design except the end, there is no way to determine *which* pulse(s) is responsible for the final sequence performance. Evaluation of a pulse sequence during design is useless to understanding the final result. This is known as 'closed-loop' design. However, by switching to an 'open-loop' formulation where the pulse sequence is randomly initialized in total and the RL algorithm is allowed to change individual pulses at a time, the reward becomes meaningful. When one pulse changes, it is clear the effect it has on the total sequence's performance. This still present a sparse reward problem in the case of modifying individual pulses - determined in small qualitative tests we performed - but this could be improved with physically relevant composite pulse choices. Fundamentally, using RL to speedup the superior physically motivated beam-search method we presented in this work is an excellent avenue for future work that brings together RL's strong sample-efficient performance and the strong performance of sequences that are similar to one another.

V. ACKNOWLEDGMENTS

I am grateful to my advisor, Professor Ronald Walsworth, for his advice on this project, funding, group meeting advice and opportunities, and opportunities to present on my work at APS March Meeting. March Meeting especially led to a number of the future work ideas presented in this thesis and helped me determine which methods had already been tried with no success. I am also grateful to Dr. Connor Hart and Dr. Kevin Olson for their help and advice on this project through many group meetings.

I am also **incredibly grateful** to Jner Tzern Oon - JJ Oon, the graduate student advisor I worked with throughout this project. From his help and work on the simulation tools employed, his advice in countless weekly meetings, incredibly insightful feedback on all the presentations, posters, talks, etc. created from this work, and overall **inspiration** he provided through the 3 years of this project, this work would absolutely not have been possible without him. I cannot be thankful enough for all the time he has spent on this project with me, and help he has provided to me personally throughout our time together.

[1] P. Peng, X. Huang, C. Yin, L. Joseph, C. Ramanathan, and P. Cappellaro, Deep reinforcement learning for quantum hamiltonian engineering, Phys. Rev. Appl. **18**, 024033 (2022).

[2] J. M. Schloss, Optimizing nitrogen-vacancy diamond magnetic sensors and imagers for broadband sensitivity, (2019).

[3] M. Schmelz, R. Stolz, V. Zakosarenko, T. Schönau, S. Anders, L. Fritzsch, M. Mück, and H.-G. Meyer, Field-stable squid magnetometer with sub-ft hz 1/2resolution based on sub-micrometer cross-type josephson tunnel junctions, Superconductor Science and Technology **24**, 065009 (2011).

[4] Y. P. Pan, J. J. Zhu, Y. Feng, Y. S. Lin, H. B. Wang, X. Y. Liu, H. Jin, Z. Wang, L. Chen, and Y. H. Wang, Improving spatial resolution of scanning squid microscopy with an on-chip design, Superconductor Science and Technology **34**, 115011 (2021).

[5] X. Gao, C. Yu, S. Zhang, H. Lin, J. Guo, M. Ma, Z. Feng, and F.-W. Sun, High sensitivity of diamond nitrogen-vacancy magnetometer with magnetic flux concentrators via enhanced fluorescence collection, Diamond and Related Materials **139**, 110348 (2023).

[6] Y. Xu, W. Zhang, and C. Tian, Recent advances on applications of nv magnetometry in condensed matter physics, Photonics Research **11**, 393 (2023).

[7] M. W. Doherty, F. Dolde, H. Fedder, F. Jelezko, J. Wrachtrup, N. B. Manson, and L. C. L. Hollenberg, Theory of the ground-state spin of the nv¡mml:math xmlns:mml="http://www.w3.org/1998/math/mathml" display="inline"¿¡mml:msup¿¡mml:mrow /¿¡mml:mo¿¡/mml:mo¿¡/mml:msup¿¡/mml:math¿center in diamond, Physical Review B **85**, 10.1103/physrevb.85.205203 (2012).
[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (The MIT Press, 2020).
[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and et al., Mastering the game of go with deep neural networks and tree search, Nature **529**, 484–489 (2016).
[10] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, Mastering diverse domains through world models (2024), arXiv:2301.04104 [cs.AI].
[11] H. H. Vallabhapurapu, J. P. Slack-Smith, V. K. Sewani, C. Adambukulam, A. Morello, J. J. Pla, and A. Laucht, Fast coherent control of a nitrogen-vacancy-center spin ensemble using a ktao$_3$ dielectric resonator at cryogenic temperatures, Phys. Rev. Appl. **16**, 044051 (2021).
[12] I. Bayn, E. H. Chen, M. E. Trusheim, L. Li, T. Schröder, O. Gaathon, M. Lu, A. Stein, M. Liu, K. Kisslinger, H. Clevenson, and D. Englund, Generation of ensembles of individually resolvable nitrogen vacancies using nanometer-scale apertures in ultrahigh-aspect ratio planar implantation masks, Nano Letters **15**, 1751–1758 (2015).
[13] H. H. Vallabhapurapu, J. P. Slack-Smith, V. K. Sewani, C. Adambukulam, A. Morello, J. J. Pla, and A. Laucht, Fast coherent control of a nitrogen-vacancy-center spin ensemble using a ktao$_3$ dielectric resonator at cryogenic temperatures, Phys. Rev. Appl. **16**, 044051 (2021).
[14] F. Dolde, I. Jakobi, B. Naydenov, N. Zhao, S. Pezzagna, C. Trautmann, J. Meijer, P. Neumann, F. Jelezko, and J. Wrachtrup, Room-temperature entanglement between single defect spins in diamond, Nature Physics **9**, 139–143 (2013).
[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning (2013), arXiv:1312.5602 [cs.LG].
[16] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, Distributional reinforcement learning with quantile regression (2017), arXiv:1710.10044 [cs.AI].
[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning (2016), arXiv:1602.01783 [cs.LG].
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms (2017), arXiv:1707.06347 [cs.LG].
[19] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory **28**, 129–137 (1982).
[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, Hindsight experience replay (2018), arXiv:1707.01495 [cs.LG].
[21] J. Shlens, Notes on kullback-leibler divergence and likelihood (2014), arXiv:1404.2000 [cs.IT].

Appendix A: RL Algorithms

1. DQN

DQN is a value-based network that generalizes the idea of Q-Learning in Reinforcement Learning [8, 15]. In Q-Learning, actions are chosen that maximize the quality of the next state. In standard Q-Learning, this is done by filling out a table with all state and action pairs. In a realistic scenario like pulse sequence design it is not feasible to fill a table with all possible state action pairs. *Deep* Q Learning learns to predict the Q value in the table for all state action pairs. Mathematically this is expressed as follows.

Given a sample from the buffer $\mathcal{D}$ (see Figure 3) s_t, a_t, r, s_{t+1}, minimize the loss

$$L = \mathbb{E}_{s_t,a_t,r,s'\sim\mathcal{D}}[(y_i - Q(s_t, a_t; \theta_i))^2] \tag{A1}$$

where y_i is the so-called 'temporal difference' target $y_i = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i)$, θ represents the model neural network parameters as usual, and Q is output by the model.

2. DQN HER

DQN HER - Hindsight Experience Replay - is a slight modification of DQN where the value function Q now depends on a sub-goal g [20]. This sub-goal g lives in the space of possible states. I.e. all possible states live in $\mathbb{S}$ and $g \in \mathbb{S}$. The goals are sampled during training to be a state that *was achieved* during the trained trajectory.

3. QRDQN

Quantile Regression DQN - QRDQN - now predicts a *distribution* for each value of Q [16]. This minor modification changes the way the action a is selected. Originally

$$a_t = \arg\max_{a_t} \theta(s_t, a_t) \tag{A2}$$

but with this distribution a_t is now

$$a_t = \arg\max_{a_t} \frac{1}{N} \sum_{j=1}^{N} \theta_j(s_t, a_t) \tag{A3}$$

where θ_j represents the model output for the jth quantile of the fixed N quantiles we are predicting.

4. A2C

Given a sampled trajectory and timestep t, collect all the rewards from t to the end of episode. For the nth step up from t, define the Return as

$$R_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}; \phi) \tag{A4}$$

where ϕ are the neural network parameters that define the value neural network for approximating V. Now define an advantage estimate $A_t = R_t - V(s_t; \phi)$ such that we can define the policy loss for the policy net θ [17]

$$L(\theta) = -\mathbb{E}_t[\log \pi(a_t|s_t;\theta)A_t] - \beta \mathbb{E}_t[\mathcal{H}(\pi(\cdot|s_t;\theta))] \tag{A5}$$

where $\mathcal{H} = -\sum_a \pi(a) \log \pi(a)$. This second $\mathcal{H}$ term controls the entropy of the policy. It encourages a high entropy policy to incentivize exploring new states.

V is fit to the return R using a simple squared error. The total loss is then the loss of the value and policy networks, which is differentiated with respect to ϕ, θ to update the two neural networks.

5. PPO

Again, like for all other models, we sample a trajectory from the buffer.

We now compute an advantage much like for A2C. We produce a generalized advantage estimate (GAE) $\hat{A}_t$ like [18]

$$\delta_t = r_t + \gamma V(s_{t+1};\phi), \hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} \tag{A6}$$

where again ϕ represents the value network, T is the sequence length, and λ, γ are control terms.

Now define for all pairs s_t, a_t

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{A7}$$

which represents the ratio of the probability of an action in a given new policy π_θ vs the old policy $\pi_{\theta_{old}}$. This is used in minimizing the policy loss

$$L = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \tag{A8}$$

with the value loss for ϕ still the standard squared loss with the return R.

Note the key difference is that the policy loss is scaled by the *change in the policy*. Therefore, large policy changes will lead to high losses and be clipped. This keeps $r_t(\theta)$ from being too high as θ is updated; this is why PPO is referred to as *proximal* policy optimization.

Also note there is an entropy bonus that is optionally added to the loss L in the same exact form as for A2C.

6. Dreamer V3

There are a large number of neural networks used in Dreamer [10]. Let's start by listing them all.

$$h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$$

$$z_t \sim q_\phi(z_t|h_t, x_t)$$

$$\hat{z}_t \sim p_\phi(\hat{z}_t, h_t)$$

$$\hat{r}_t \sim p_\phi(\hat{r}_t|h_t, z_t)$$

$$\hat{c}_t \sim p_\phi(\hat{c}_t|h_t, z_t)$$

$$\hat{x}_t \sim p_\phi(\hat{x}_t|h_t, z_t)$$

The input x_t is encoded into a latent state z_t using the encoder function q_ϕ. This encoder function is trained using the decoder head of p_ϕ. p_ϕ produces the decoded value $\hat{x}_t$ which should match the originally encoded state x_t. With the encoded state z_t, a sequence model f_ϕ predicts a sequence of z, h vectors. z represents the latent space encoding of what *would* be the state at the respective future timestep, and h is a state vector that contains information and values necessary to the model for long-term prediction (entirely learned). The continue flag and reward predictors $\hat{c}_t$ and $\hat{r}_t$ are used to ensure that the world model is encoding the state at timestep t, x_t correctly so the long term dynamics predicted by $f, \hat{z}$ are accurate.

The model then optimizes the following loss

$$L = \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T} \alpha L_{pred} + \beta L_{dyn} + \chi L_{rep}\right] \tag{A9}$$

where α, β, χ are hyperparameter control terms and T is the sequence length. These losses are then respectively

$$L_{pred} = -\ln p_\phi(x_t|z_t, h_t) - \ln p_\phi(r_t|z_t, h_t) - \ln p_\phi(c_t|z_t, h_t) \tag{A10}$$

$$L_{dyn} = \max(1, \text{KL}\left[\text{sg}(q_\phi)||p_\phi(z_t|h_t)\right]) \tag{A11}$$

$$L_{rep} = \max(1, \text{KL}\left[q_\phi||\text{sg}(p_\phi(z_t|h_t))\right]) \tag{A12}$$

where sg is the stop-gradient operator and KL is the KL difference; the KL divergence qualitatively is an effective distance function between two probability distributions [21].

This ensures that the predictions of the world model are correct. To use this in practice, the model starts with the initial RL real state (pulse sequence) s_0, then uses the world model to dream out a trajectory of states, actions, etc. We then perform actor-critic learning, similar to work on the A2C algorithm, on the *dreamed* trajectory. This ensures the agent learns how to work efficiently in its increasingly accurate dreamed space. So as time goes-on, the model learns how to maximize its return in the Dreamer latent space (the encoded space).