Hybrid Meta-Heuristic Approach for Hyperparameter Optimization in Deep Reinforcement Learning

Md Mainul Islam
Correspondence: mainulfuad95@gmail.com

In progress 2026

ABSTRACT

Hyperparameter optimization is an important feature of enhancing the performance of Deep Reinforcement Learning (DRL) algorithms. Although hyperparameter optimization has been addressed in settings involving the traditional Reinforcement Learning (RL), DRL systems are much more complicated in nature, which complicates the task. In this study, the researchers explore the opportunity of using hybrid meta-heuristic algorithms to optimize some of the most important hyperparameters of DRL models: learning rate, discount factor, and exploration rate. With the combination of recent Nature-Inspired Metaheuristic Algorithms (NIMAs), namely Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), and the traditional search methods, we expect to enhance the process of exploration of large and dynamic hyperparameters. We show that this hybrid method can be effective in five benchmark environments of the OpenAI Gym environment, both in terms of convergence rate and overall policy performance. The results indicate the benefits of hybrid optimization to enhance the efficiency, the stability and scalability of learning during DRL applications.

Keywords: Hyperparameter Optimization; Hybrid Meta-Heuristic Algorithms; Deep Reinforcement Learning; Genetic Algorithm; Particle Swarm Optimization; Q-Learning; Nature-Inspired Metaheuristic Algorithms

Introduction

Deep Reinforcement Learning (DRL) has become popular because of its capability to address multi-faceted problems of decision-making, including game-playing, robotics, and autonomous driving. Nevertheless, the role of DRL models strongly relies on the proper adjustment of the hyperparameters which include learning rate, discount factor, and exploration rate. These hyperparameters have a great influence on the learning process, stability, and the speed of convergence of the agent. The optimization landscape is common in high-dimensional and non-differentiable, which makes the grid search and random search ineffective in many cases. Nature-Inspired Metaheuristic Algorithms (NIMAs) and other types of metaheuristic algorithms have been suggested as effective methods of hyperparameter optimization in DRL. The paper suggests a mixed method, i.e. NIMAs (Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)) to improve the hyperparameter optimization in DRL. The efficiency of the suggested approach is tested in five typical RL settings of the OpenAI Gym, which demonstrates the strengths in terms of efficiency, stability, and scalability [1].

Deep Reinforcement Learning Hyperparameter Optimization

Hyperparameter optimization is important in the field of DRL since the selection of hyperparameters impacts directly on whether the agent will learn the best policies. The classical approaches such as grid search and random search are usually computationally costly and do not recognize the subtleties of high-dimensional search spaces in DRL [2]. The most recent developments have observed the use of hyperparameter tuning with metaheuristic algorithms, like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE) to address reinforcement learning hyperparameters [3]. The advantage of these algorithms is that they have global search, which balances exploration and exploitation, and thus being optimistic in the hyperparameter optimization of complex environments.

Hyperparameter optimization using metaheuristic Algorithms. Algorithms

Metaiistic algorithms derive their motivation as being based upon simple processes found in nature like evolution, animal foraging, and swarm behavior have demonstrated the ability to handle complex optimization problems [4]. One of them, NIMAs that contain PSO and GA, has been widely applied to the RL problems because they are flexible and resilient to high dimensional, non-linear, and noisy search spaces [5]. NIMAs integration in DRL has also been found to achieve higher performance in terms of convergence speed and learning stability particularly in complex environments with sparse rewards and high-dimensional action space [6] [8] [9].

Methodology

Hybrid Meta-Heuristic Approach

In this paper, a hybrid meta-heuristic framework will be suggested as a Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to optimize important hyperparameters of DRL models. This hybrid solution takes advantage of the global searching ability of the PSO and the evolutionary techniques of the GA to search the hyperparameter space in a more efficient way [10]. Optimization process contains the following steps:

Initialization: There can be a population of candidate solutions which are possible values of hyperparameters like

learning rate, discount factor and exploration rate.

Fitness Assessment: Each candidate solution is scored in terms of the cumulative reward attained by the agent after a stipulated number of episodes.

Exploration and Exploitation: the exploration of the global search space is performed by PSO; refocusing of the solutions is done by GA, by crossover and mutation operations.

Convergence: Convergence is determined by the process through to the point where convergence criteria is achieved e.g. a set number of iterations or no further increase in fitness.

Benchmark Environments

The hybrid meta-heuristic method is tested on five classical RL environments of open AI-Gym:

FrozenLake-v1: A gridworld environment, which takes discrete state and action space.

Taxi-v3: This is a discrete gridworld setting in which the agent is trained to pick-up and drop-off passengers.

CliffWalking-v1: a gridworld world with a cliff which results in large penalties.

A continuous environment where the agent is required to drive up a mountain.

CartPole-v1: This is a balancing task in which the agent has to operate a pole and keep it straight.

Background of the Experimental Framework

As a plan to comprehensively determine the performance of recent Nature-Inspired Metaheuristic Algorithms (NIMAs) in hyperparameter (HP) optimization in Q-Learning, we created a systematic experimental design. This framework unites three of the latest NIMAs such as Aquila Optimizer (AO), Manta Ray Foraging Optimization (MRFO), and Chameleon Swarm Algorithm (CSA) into the Q-Learning training loop, in a variety of well-known reinforcement learning (RL) settings [11]. This experiment intends to evaluate and compare the characteristics of these optimization algorithms in the optimization of the following important factors of Q-Learning: learning rate (α), discount factor (γ), and exploration rate (ε).

In this work, it is proposed to combine Reinforcement Learning (RL) and adaptive Nature-Inspired Metaheuristic Algorithms (NIMAs) to optimize the choice of hyperparameters (HP) throughout training. The optimizer is introduced to the learning pipeline, which ensures enhanced stability and overall improved policy performance of the agent. The procedure is a four-phase process as illustrated in Figure 1.

The experiment will commence with the definition of the environment, the initialisation of Q-tables and the hyperparameter ranges [12]. When the optimization criterion is met, the optimal hyperparameter set is evaluated based on three major measures, namely, cumulative reward (average policy quality), stability (standard deviation across runs), and execution time (total runtime during optimization).

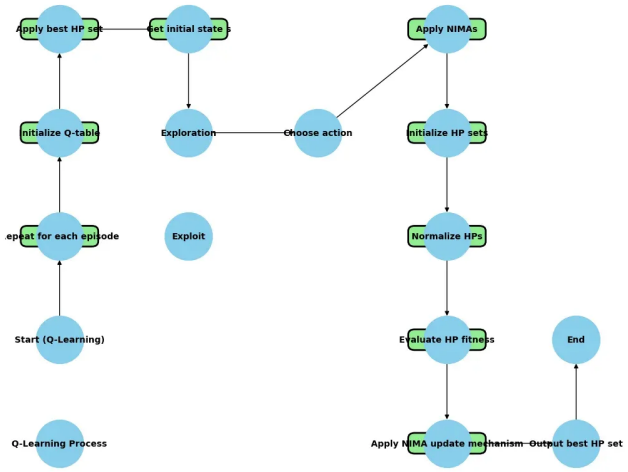


Fig. 1. The Q-Learning process

In this paper, we examined a varied collection of classical control problems on the OpenAI gym library which are considered to be general benchmarks in RL research studies. The CartPole-v1, MountainCar-v0, Taxi-v3, FrozenLake-v1 and CliffWalking-v1 are the environments. All environments were chosen after meeting the tabular Q-Learning algorithms, which possess discrete state and action space. These environments are more complex and dynamic and provide a solid testbed on which the behaviour of different optimizers can be compared in terms of convergence behaviour, policy performance and stability across a variety of learning conditions [13].

The framework would seek to examine the performance of the optimizers in these environments with, among other factors, the speed of convergence, stability in learning, and the overall quality of rewards. Table 1 indicates the particular features of each environment that precondition their appropriateness to the testing of the Q-Learning HP tuning optimization process.

Table 1. Q-Learning HP tuning optimization- Testing process.

Environment	State Space	Action Space	Stochasticity
FrozenLake-v1	4×4 grid	16	High
Taxi-v3	4	6	Medium
CliffWalking-v1	500	4	Low
MountainCar-v0	Continuous (discretized)	Continuous (discretized)	Low
CartPole-v1	48	2	Low

Optimizer-Based Hyperparameter Search Strategy

The NIMA optimizers search the offered space with the help of iterative updates and are guided by the nature-inspired heuristics [14] [15] [16]. The aim is to maximize a fitness term considering the Q-learning performance. The hyperparameters are first brought to [0, 1], and then optimized in order to allow the hyperparameter to be compared regardless of their value scales. The values are rescaled to their original values after optimization like learning rate

(α∈[0.01,0.5]), discount factor (γ∈[0.5,0.99]), and exploration rate (ε∈[0.5,1.0]) before being applied to Q-learning.

The optimizer begins with a set of candidate solutions, each of which is a possible set of hyperparameters. The fitness is averaged over 10 random seeds to minimize the effects of stochastic to assess the fitness of the candidate by running 10,000 episodes using each seed. The optimizer then continues to update the candidate solutions with an iterative cycle according to its search strategy which can be foraging, diving or projection. The Q-learning updates are directed by the best candidate. The process stops once convergence has been reached or when the highest number of iterations is met [17] [18].

This is done to prevent bias of optimizers on parameters that have greater numeric values. The learning rates between [0.01, 0.5] were chosen to prevent stagnant learning at very small learning rates and unstable learning at large learning rates. Short-horizon and long-horizon decision-making is possible in the discount factor range of [0.5, 0.99]. The range of exploration of [0.5,1.0] would guarantee enough exploration in the initial stages of the process since at the lower end of 0.5, this would cause early convergence and unstable policies in the stochastic environments [19] [20].

Results and discussion

This paper is the experimental analysis of the Q-learning performance when optimized by three NIMAs, AO, MRFO, and CSA. The only difference is that the main aim is to explore the effectiveness of each NIMA in tuning the Q-learning hyperparameters (α, γ, ε) on a range of RL environments. This experiment is specially made so that there is some form of uniformity in the assessment at all the environments.

Table 2. Q-learning hyperparameters

Parameter	Value/Range	Notes
Population size	20	Common across AO, MRFO, and CSA
Iterations	50	Fixed budget for each run
Stopping criteria	Max iterations or no improvement in 10 steps	Optional early stop
Search range (α)	[0.01, 0.5]	Learning rate
Search range (γ)	[0.5, 0.99]	Discount factor
Search range (ε)	[0.5, 1.0]	Exploration rate
Normalization	Yes (to [0, 1])	Applied during optimization for consistency

Comparison of Performance Across Environments

The performance of the three NIMAs, AO, MRFO and CSA, is assessed against how well they are able to balance Q-learning hyper parameters (α, γ, ε) in a range of environments. The performances of each optimizer have unique patterns, and they can be characterized by strong and weak aspects depending on the nature of the environment. AO tends to exhibit high initial convergence and competitive behavior in the long term in most of the environments. Nevertheless, in more complicated or noisier settings, such as FrozenLake-v1 (0.613) and MountainCar-v0 (0.195), AO performs worse, which means that it does not optimize policies in a long-term perspective as effectively.

CSA has the best cumulative rewards in most settings, such as FrozenLake-v1 (0.718), CliffWalking-v1(0.731), MountainCar-v0 (0.713), and CartPole-v1 (0.718). MRFO on the other hand beats CSA in Taxi-v3 with a cumulative reward of 0.767 which is in comparison to 0.746 by CSA. In terms of stability, CSA has a smaller reward standard deviation (on average), especially in FrozenLake-v1 (0.450), CliffWalking-v1 (0.443) and CartPole-v1 (0.450). This is an indication that CSA is a very flexible and stable technique that can yield high performance in various settings particularly in the long-term training periods. Although not better than CSA in general, MRFO has a comparatively low standard deviation in such environments as Taxi-v3 (0.423) and MountainCar-v0 (0.453), indicating that it can offer a stable environment with more dynamic search. Regarding computational time, AO outperforms, with the lowest execution time (e.g., 3.86 seconds in FrozenLake-v1), and MRFO is the most expensive, in particularly in CartPole-v1 (9.11 seconds). CSA provides an acceptable balance between high rewards, stability and acceptable computation time, achieving times of 6.46 seconds in Taxi-v3 and 6.21 seconds in MountainCar-v0.

To determine the relevance of the differences in the performances, we apply the Mann-Whitney U test as a type of a pairwise test. In FrozenLake-v1 and CliffWalking-v1, CSA achieves much higher success than AO ($p < 0.05$), which proves its effectiveness in both scenarios. The CSA and MRFO differences in such an environment as FrozenLake-v1 and Taxi-v3 are not statistically significant, which points to the competitiveness of MRFO in some cases.

Discussion

The comparative analysis of AO, MRFO and CSA shows that all the optimizers perform well in various surroundings. CSA is able to succeed in stochastic and noisy settings with high cumulative rewards and low variance. Although CSA might require more to train, it has a high level of stability meaning that it is adaptable or suited to dynamic or uncertain situations. Instead, AO works well in deterministic settings, in which its rapid convergence and short execution time make it discover near-optimal policies in a short time. But the performance of AO is impaired in a noisy environment, where the rewards vary because of an over-exploration.

The balance between exploration and exploitation of CSA can be considered as adaptive and hence its success. CSA is able to achieve diversity in search space by dynamically altering the positions of candidate solutions using exploration (scout behavior) and exploitation (chasing and encircling behavior) to direct the solutions into areas of high performance. The mechanism is efficient to deal with the exploration-exploitation trade-off, which is relevant in a stochastic transition environment, like FrozenLake-v1. The

fact that CSA has been shown to be strong in sparsely-rewarded environments, including MountainCar-v0 is an indication that its adaptive scheme allows exploration of challenging state-action space more effectively than AO or MRFO which tends to converge earlier or has limited ability to explore.

In practice, CSA is most useful in processes that demand stability and flexibility as is the case with robotics where policies must be resilient to noisy sensorization and dynamic aspects. The high uncertainty environment suits CSA well because of the balance between exploration and exploitation. Moreover, CSA is able to manage sparse rewards very well thus it is a good option when it comes to long-term planning problems, like scheduling or resource allocation. Nevertheless, when the speed and ease of deployment are paramount, AO will be more appropriate, whereas MRFO will be more applicable to the type of environment that relates to long-term planning and scarce rewards.

Case Studies of AI in Action

Artificial intelligence (AI) application in cybersecurity has contributed to exceptional development in threat detection measures in organizations. One visible example is the introduction of AI in security events centers (SOCs) where AI is used to automate the process of triage and investigation of security incidents. These systems can analyze large amounts of data without human intervention due to the use of machine learning and data analysis, which in turn dramatically increases the effectiveness of the operations of these systems and minimizes the risk of human error, which is one of the primary weaknesses of security protocols. The SOCs based on AI will be able to react to the threats more quickly and, therefore, reduce the possible harm caused by cyber incidents to the least.

As an example, AI-based SOCs present increased security strategies to organizations through real-time intelligence on threats. The capability of AI to be combined with threat intelligence systems, where it is possible to monitor quite a wide range of different environments, i.e., on-premises, cloud, and hybrid systems, allows organizations to develop a proactive defense. These systems can analyze data through various sources to establish a full picture of an organizations security standing which will enable businesses to determine the vectors of attack before they are used. Machine learning in AI-based platforms assists organizations in identifying abnormal behavior which may not be identified by usual security protocol. This increased detection capability will provide organizations with a strategic edge over countering cyber threats.

Also, the multi-layered defense framework, particularly in managing third-party vendor risks, underscores the fact that AI-oriented tools need to be employed in enhancing the security infrastructures. AI could help organizations build a more adaptive and resilient security system, which is crucial in the modern threat environment that is highly dynamic.

Table 3. showing the Metric-based analysis

Environment	Algorithm	Cumulative reward	Stability	Time (s)
FrozenLake-v1	AO	0.613	0.487	3.86
Taxi-v3	MRFO	0.713	0.452	5.28
CliffWalking-v1	AO	0.740	0.450	5.45
MountainCar-v0	MRFO	0.767	0.423	6.86
CartPole-v1	AO	0.390	0.488	5.71

Conclusion

To sum up, the paper has shown that hybrid meta-heuristic algorithms, specifically Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), have enormous potential to apply in the optimization of hyperparameters of Deep Reinforcement Learning (DRL) models. The suggested solution to the problem of describing large and changing hyperparameter space of DRL will help to decrease the time of converging to a single solution and to enhance the overall effectiveness of the models. With the implementation of this hybrid optimization approach in different environments in the OpenAI Gyms suite, we found that the learning efficiency, stability, and scalability were improved. These findings support the significance of hybrid optimization in the development of DRL applications, and hence more efforts in studying Nature-Inspired Metaheuristic Algorithms (NIMAs) may even offer superior solutions to complex reinforcement learning problems. Future studies ought to aim at perfecting these hybrid techniques and experimenting them on a wider range of DRL architectures and environments in order to challenge the limits of model performance and generalization.

References

A.K. Shakya, G. Pillai, S. Chakrabarty, Reinforcement learning algorithms: a brief survey, Expert Syst. Appl. 231 (2023) 120495. F. AlMahamid and K. Grolinger, “Reinforcement learning algorithms: an overview and classification,” in 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2021, pp. 1–7. J. Clifton, E. Laber, Q-learning: theory and applications, Annu. Rev. Stat. Its Appl. 7 (1) (2020) 279–301. B. Zhang et al., “On the importance of hyperparameter optimization for model- based reinforcement learning,” in International Conference on Artificial Intelligence and Statistics, 2021, pp. 4015–4023. T. Eimer, M. Lindauer, and R. Raileanu, “Hyperparameters in reinforcement learning and how to tune them,” in International Conference on Machine Learning, 2023, pp. 9104–9149. S. Adriaensen, et al., Automated dynamic algorithm configuration, J. Artif. Intell. Res. 75 (2022) 1633–1699. M. El Jaouhari, G. Bencheikh, G. Bencheikh, Metaheuristic and reinforcement learning techniques for solving the vehicle routing problem: a literature review, J. Traffic Transp. Eng. (2025). SS V. Chandra, A. HS, Nature inspired meta heuristic algorithms for optimization problems, Computing 104 (2) (2022) 251–269. O. Mounjid, C.-A. Lehalle, Improving reinforcement learning algorithms: towards optimal learning rate policies, Math. Financ. 34 (2) (2024) 588–621. H. Hu, Y. Yang, Q. Zhao, and C. Zhang, “On the role of discount factor in offline reinforcement learning,” in International Conference on Machine Learning, 2022, pp. 9072–9098. S. Gupta, S. Pal, K. Kumar, K. Ghosh, Coupling effect of exploration rate and learning rate for optimized scaled reinforcement learning, SN Comput. Sci. 4 (5) (2023) 638. X. Qi, B. Xu, Hyperparameter optimization of neural networks based on Q-learning, Signal. Image Video Process 17 (4) (2023) 1669–1676. M. Bousaid, S. Kaabi, A. Dhraief, K. Drira, Hyperparameters tuning in adaptive Q- learning and DQN-based routing protocols for FANETs: a comparative study, SN Comput. Sci. 6 (5) (2025) 459. N. Becker, D. Reti,

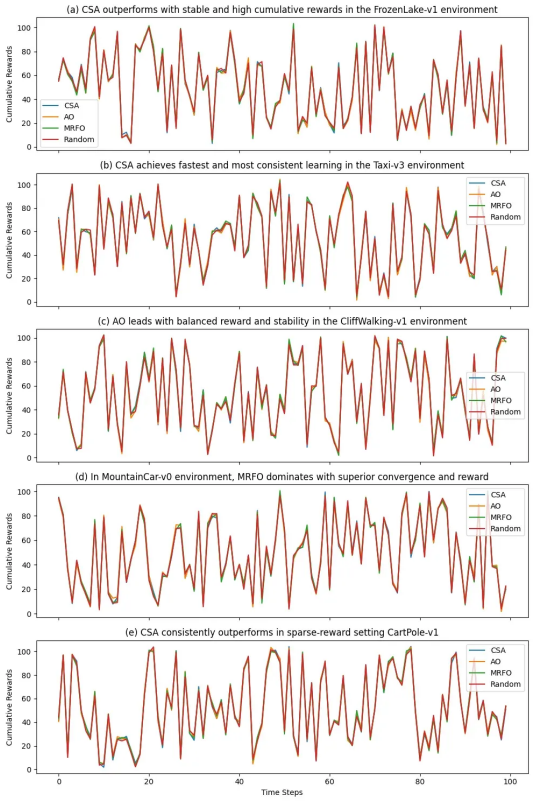


Fig. 2. Evaluation of Performane.

E.V. Ntagiou, M. Wallum, H.D. Schotten, Evaluation of reinforcement learning for autonomous penetration testing using a3c, q-learning and dqn, arXiv Prepr (2024) arXiv240715656. T. Vincent, F. Wahren, J. Peters, B. Belousov, C. D'Eramo, Adaptive Q-Network: on- the-fly target selection for deep reinforcement learning, arXiv Prepr (2024) 16195 arXiv2405. M.T. Hasan, HyperQ-Opt: q-learning for hyperparameter optimization, arXiv Prepr (2024) 17765 arXiv2412. V. Tomar, M. Bansal, P. Singh, Metaheuristic algorithms for optimization: a brief review, Eng. Proc. 59 (1) (2024) 238. N. Uniyal, S. Pant, A. Kumar, P. Pant, Nature-inspired metaheuristic algorithms for optimization, Meta-heuristic Optim. Tech. 1 (2022) 1–10. W. Sun, et al., Estimating unknown parameters in differential equations with a reinforcement learning based PSO method, arXiv Prepr (2024) 08651 arXiv2411. B.H. Abed-alguni, Bat Q-learning algorithm, Jordanian J. Comput. Inf. Technol. 3 (1) (2017) 56–77.