Using Deep Q-Network for Selection Hyper-Heuristics

Augusto Dantas
Federal University of Paraná
Paraná, Brazil
aldantas@inf.ufpr.br

Alexander Fiabane do Rego
Federal University of Paraná
Paraná, Brazil
afrego@inf.ufpr.br

Aurora Pozo
Federal University of Paraná
Paraná, Brazil
aurora@inf.ufpr.br

ABSTRACT

Hyper-Heuristics is an active research field that aims to automatically select (or generate) the best low-level heuristic in each step of the search process. This work investigates a Hyper-Heuristic with a Deep Q-Network (DQN) selection strategy and compares it with two state-of-the-art approaches, namely the Dynamic MAB and the Fitness-Rate-Rank MAB. The experiments conducted on two domains from the HyFlex framework showed that the DQN approach outperformed the others on the Vehicle Routing Problem and was competitive on the Traveling Salesman Problem. This indicates that the DQN is a robust selection strategy that is less sensitive to the domain than the MAB based approaches.

CCS CONCEPTS

• **Computing methodologies** → *Reinforcement learning*; **Search methodologies**;

KEYWORDS

Hyper-Heuristic, Reinforcement Learning, Combinatorial Optimization

ACM Reference Format:
Augusto Dantas, Alexander Fiabane do Rego, and Aurora Pozo. 2021. Using Deep Q-Network for Selection Hyper-Heuristics. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3449726.3463187

1 INTRODUCTION

Real-world complex optimization problems often rely on heuristic approaches to achieve a good feasible solution in a reasonable computational time [2]. However, their performance heavily depends on problem-specific configurations, meaning that a domain knowledge is required for adapting them on new applications [17].

Because of that, in the past decade many researchers turned their attention to Hyper-Heuristics (HH) [6]. Besides offering competitive results against problem-specific heuristics, HHs also aim at being generalized approaches that deliver good solutions across different domains [3].

As a search methodology, HHs explore the search space of low-level heuristics (e.g., evolutionary operators) [3]. Throughout an iterative process, at each step of the search, a selection Hyper-Heuristic chooses a heuristic and applies it to the current solution. Then, it accepts or rejects the new solution according to an acceptance criteria. To avoid getting stuck into local optima solutions, good HHs must know which is the appropriate low-level heuristic to explore a different area of the search space at the time [3].

According to Burke et al. [3], HHs can be classified by its source of learning feedback, which can be online, offline, or no-learning. This work focuses on online approaches, which are usually achieved by the use of Reinforcement Learning (RL) or meta-heuristic algorithms as high level search engine [3].

There are several RL based selection strategies for Hyper-Heuristics in the literature. Some of them use the received feedback to update a probability vector that controls the selection (e.g., Probability Matching (PM) and Adaptive Pursuit (AP) [8]). Others are based on selection rules that aim at tackling the exploration versus exploitation dilemma (e.g., Choice Function and Multi Armed Bandit based strategies [8]).

In this work, we investigate a selection Hyper-Heuristic that uses a Deep Q-Network to choose the heuristics. The selection agent is updated while solving an instance using the Q-learning algorithm [16] with an Artificial Neural Network as function approximator [14]. In this way, we model the task of selecting the low-level heuristics as a Markov Decision Process (MDP) [12], which implicates that the decision of the agent is made based on the current observed state representation. The advantage of using a state based approach is that the selection strategy can learn a pattern that identifies the proper heuristics for the current search behavior, instead of only relying on the current operator probabilities.

We compare this approach with two state-of-the-art MAB based selection rules, namely the Dynamic MAB (DMAB) [4] and the Fitness-Rate-Rank MAB (FRRMAB) [10]. The MAB problem can be seen as a special case of Reinforcement Learning with only a single state [14], thus contrasting with the MDP state representation. We perform experiments using the problems and heuristics from the Hyper-Heuristics Flexible framework (HyFlex) [11].

There is a large literature on the use of MAB strategies for selection HHs. In an investigative approach, Almeida et al. [5] compared three MABs: FRRMAB, Restless MAB, and Contextual MAB. They considered four crossovers and two mutations operator of the genetic algorithm and performed tests on the bi-objective Permutation Flow Shop Problem. Similarly, Ferreira et al. [7] compared the Sliding Window MAB and the FFRMAB with their proposal variant Fitness-Rate-Average MAB. They conducted the experiments on the HyFlex problem domains, presenting competitive results. MAB based strategies are also employed as the selection mechanism on more elaborate HH algorithms. Soria-Alcaraz et al. [13] proposed

GECCO '21 Companion, July 10–14, 2021, Lille, France
© 2021 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463187

an Iterated Local Search (ILS) based HH that uses the Dynamic MAB to select the move operator during the local search phase. The reported results on the VRP domain from HyFlex demonstrated that the DMAB is a powerful selection strategy for HHs.

Moreover, there are a few works that define a state representation for a Reinforcement Learning based HH. Handoko et al. [9] defined a discrete state space that relates to fitness improvement and diversity level. Then, a tabular Q-learning [16] is used to update the state-action values and select among the crossover operator of a evolutionary algorithm. The experimental results demonstrate that this approach is competitive with classical credit assignment mechanisms (AP, PM and MAB), while being less sensitive to the number of operators. Similarly, Teng et al. [15] defined a continuous state space that includes additional information about the population and its offspring. Then, a Self-organizing Neural Network is trained offline to select the crossover operator. The performance of this approach was competitive with other selection mechanisms and even better on some instances, thus highlighting the advantages of using a state-based selection mechanism. In contrast to these works, this paper investigates the use of an online learning selection agent through Q-learning, but with a continuous state representation.

The remainder of this paper is organized as follows: in Sect. 2 we describe the selection strategies performed during the experiments, including our DQN-based approach. The experimental setup and results are given in Sects. 3 and 4, respectively. Finally, we draw some conclusions and indicate future works in Sect. 5.

2 SELECTION STRATEGIES

This section details the three selection strategies compared in this work. Both MAB strategies use the same selection rule (Upper Confidence Bound [1]). The state representation of DQN is based on the sliding window strategy from FRRMAB.

2.1 Dynamic Multi-Armed Bandit

A MAB framework is composed of N arms (e.g., operators) and a selection rule for selecting an arm at each step. The goal is to maximize the cumulative reward gathered over time [13]. Among several algorithms to solve the MAB, the Upper Confidence Bound (UCB) [1] is one of the most known in the literature, as it provides asymptotic optimality guarantees. The UCB chooses an action based on the following rule

$$p_{i,t} + C\sqrt{\frac{2log(\sum_{j=1}^{N} n_{j,t})}{n_{i,t}}} \tag{1}$$

where $n_{i,t}$ is the number of times the ith arm has been chosen, and $p_{i,t}$ the average reward it has received up to time t. The scaling factor C gives a balance between selecting the best arm so far ($p_{i,t}$, i.e., exploitation) and those that have not been selected for a while (second term in the Eq. 1, i.e., exploration).

However, the UCB algorithm was designed to work in static environments. This is not the case in the Hyper-Heuristic context, where the quality of the low-level heuristics can vary along the HH iterations [8]. Hence, the Dynamic MAB, proposed by [4], incorporates the *Page-Hikley* (PH) statistical test to deal with this issue. This mechanism resembles a context-drifting detection, but is related to the performance of the operators throughout the execution of the algorithm. Once a change in the reward distribution is detected, according to the PH test, the DMAB resets the empirical value estimates and the confidence intervals (p and n in Eq. 1, respectively) of the UCB [4].

2.2 Fitness-Rate-Rank Multi-Armed Bandit

The Fitness-Rate-Rank MAB [10] proposes the use of Fitness Improvement Rate (FIR) to measure the impact of the application of an operator i at time t, which is defined as

$$FIR_{i,t} = \frac{pf_{i,t} - cf_{i,t}}{pf_{i,t}} \tag{2}$$

where $pf_{i,t}$ is the fitness value of the original solution, and $cf_{i,t}$ is the fitness value of the offspring.

Moreover, the FFRMAB uses a sliding window of size W to store the indexes of past operators, and their respective FIRs. This sliding window is organized as a First-in First-out (FIFO) structure and reflects the state of the search process. Then, the empirical reward $Reward_i$ is computed as the sum of all FIR values for each operator i in the sliding window.

In order to give an appropriate credit value for an operator, the FRRMAB ranks all the computed $Reward_i$ in descending order. Then, it assigns a decay value to them based on their rank value $Rank_i$ and on a decaying factor $D \in [0, 1]$

$$Decay_i = D^{Rank_i} \times Reward_i \tag{3}$$

The D factor controls the influence for the best operator (the smaller the value, the larger influence). Finally, the Fitness-Rate-Rank (FRR) of an operator i is given by

$$FRR_{i,t} = \frac{Decay_i}{\sum_{j=1}^{N} Decay_j} \tag{4}$$

These $FRR_{i,t}$ values are set as the value estimate $p_{i,t}$ in the UCB equation (1). Also, the $n_{i,t}$ values considers only the amount of time that the operator appears in the current sliding window. This differs from the traditional MAB and other variants such as the DMAB, where the value estimate $p_{i,t}$ is computed as the average of all rewards received so far.

2.3 Deep Q-Network

The classic Q-learning algorithm keeps a table that stores the Q-values (the estimate value of performing an action at current state) of all state-action pairs [16]. This table is then updated accordingly to the feedback the agent receives upon interacting with the environment. However, in a continuous state space, keeping the Q-table is not feasible due to the high dimensionality of the problem [14]. Instead, we can use a function approximation model (called the Q-model) that gives the estimate Q-values. In DQN, the Q-model is defined by an Artificial Neural Network (ANN), in which the inputs are the current observed state representation, and the output layer yields the predicted Q-values for the current state-action pairs.

With these estimate Q-values, the agent selects the next action (low-level heuristic) according to its exploration policy. We used the ϵ-greedy policy, that selects a random action with probability ϵ, and selects the action with the highest Q-value with probability

$1 - \epsilon$. Thus, ϵ is a parameter that controls the degree of exploration of the agent and is usually set to a small value [14].

After performing the action, receiving the reward and observing the next state, the Q-model is updated by running one iteration of gradient descent on the Artificial Neural Network, with the following target value

$$\text{target} = \text{reward} + \gamma \max_{a'} Q\left(s', a'\right) \tag{5}$$

where s' is the next state after performing the action, and $\max_{a'} Q\left(s', a'\right)$ is the highest Q-value of all possible actions from state s'. The discount factor γ controls the influence of the future estimate rewards.

We defined the state representation as the normalized average rewards of each operator. For this, we used the same sliding window structure from FRRMAB. Hence, if we have 10 available low-level heuristics, for example, the state is represented as a vector of 10 values ranging [0,1]. The idea is to investigate if the past observed rewards can be representative enough to allow the DQN to learn a proper selection policy.

3 EXPERIMENTAL SETUP

We employed the three selection strategies under a standard selection Hyper-Heuristic algorithm, as shown in Algorithm 1. Iteratively, it selects and applies a low-level heuristic on the current solution and computes the reward. Then, the acceptance criteria decides if the new solution is accepted and, at last, the HH calls the update method of the corresponding selection model.

Algorithm 1: Selection Hyper-Heuristic

```
Input: A initial solution $\phi$ with size $n$
Output: The best found solution
repeat
    heuristic ← SelectHeuristic()
    $\phi'$ ← ApplyHeuristic($\phi$, heuristic)
    reward ← GetReward(f($\phi$), f($\phi'$))
    if AcceptSolution($\phi'$) then
        $\phi \leftarrow \phi'$
    end
    UpdateSelectionModel(reward)
until stopping criteria is not met
```

The reward is defined as the FIR value (Eq. 2) and was kept the same for all selection strategies. Since our goal is to investigate the learning ability of the selection strategies, the acceptance criteria accepts all solutions. In this way, the actions of the agent always reflects a change in the environment.

We conducted the experiments on two problems from the HyFlex Framework [11], the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). The HyFlex provides 4 types of low-level heuristics for each problem: mutational, ruin-and-recreate, local search and crossover. We included all of them, except the crossover group, into the selection pool. In this way, there are 9 heuristics available for the TSP, and 8 for the VRP. We refer to the documentation for more details about these heuristics [11].

We executed each selection strategies 31 times on every instance with different random seeds (10 TSP and 10 VRP instances). We set the stopping criteria as 300 seconds of CPU running time on a Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz. The source-code and the data (such as the parameters configuration) will be available.

4 RESULTS AND DISCUSSION

We compared the results obtained by each approach on every instance using the Friedman hypothesis test and a pairwise post-hoc test with the Bergmann correction. Figure 1 shows that the DQN selection strategy outperformed the others with statistical difference (at 95% confidence level). Meanwhile, both MAB strategies were statistically equivalent considering all VRP and TSP instances.

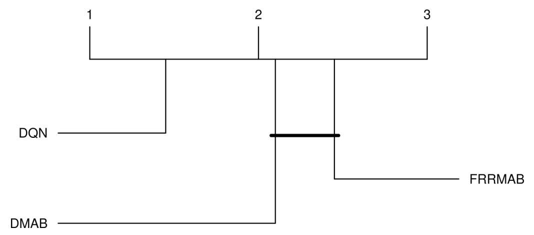


Figure 1: Friedman ranking with pairwise statistical test ($p < 0.05$). The smaller the rank, the better. The connected lines indicate the approaches that are statistically equivalent

Additionally, we also compared their performance by individual instances. Table 1 reports the average and standard deviation of the best solution found by each selection strategy in the 31 runs. Bold values indicate that the corresponding approach achieved a better performance with statistical difference, and gray background highlights all approaches that were statistically equivalent to the approach with the best rank at that instance. For this, we applied a Kruskal-Wallis hypothesis test on the whole sample, followed by a pairwise Dunn's test using the approach with the best rank (lowest average) as the control, considering a confidence level of 95%.

As we can observe, the DQN selection strategy was the best approach on the VRP instances, outperforming the others on 9 out of 10 instances. Meanwhile, for the TSP instances the DMAB strategy achieved better results. However, the DQN was still able to beat it on 1 instance and to be statistically equivalent on 3 other instances. This shows that the DQN can be more stable regarding the problem domain.

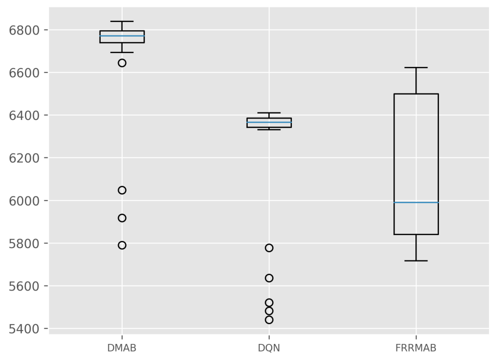
In fact, the DQN achieved results with less variant distributions on most instances. Figure 2 shows the distribution of the found solutions by all approaches on some selected instances. Figure 2a shows that even on the VRP instance where the DQN did not outperform the FRRMAB, it presented less variance. Figure 2b highlights the big advantage that DQN achieved on the larger VRP instances. Meanwhile, Figures 2c and 2d shows that the DQN were also more stable on the TSP instances where it was better or equivalent to the MAB approaches.

5 CONCLUSION AND FUTURE WORKS

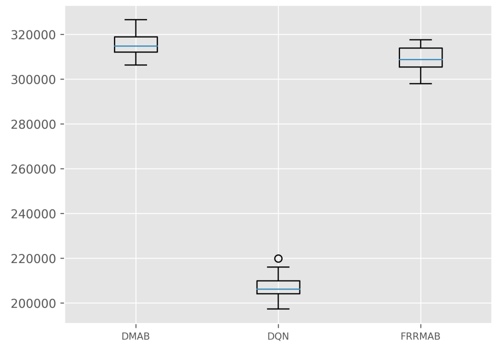
This paper investigates the use of Deep Q-Network for selecting heuristics under a simple Hyper-Heuristic algorithm. We compared this approach with two state-of-the-art selection strategies based on Multi-Armed Bandit. We conducted experiments on two problem domains from the HyFlex Framework: Traveling Salesman Problem and Vehicle Routing Problem. The results showed that the DQN was able to outperform the others on the VRP and be competitive on the TSP, thus highlighting the stability of the approach.

Table 1: Average (and standard deviation) results of each approach on every instance. Gray cells indicate statistical equivalence ($p < 0.05$) and bold values mean the approach outperformed the others

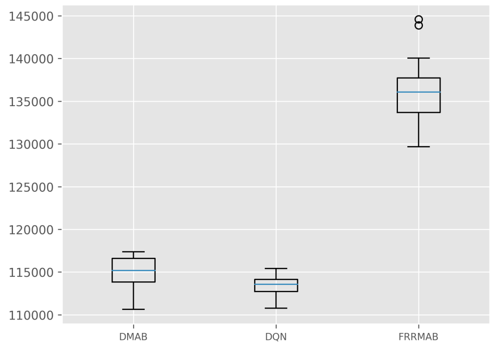
Instance	DMAB	DQN	FRRMAB
VRP-R106	18197.53 (505.2)	**15218.17 (361.84)**	17364.95 (470.25)
VRP-R201	6687.58 (256.58)	6242.36 (299.0)	6136.91 (327.37)
VRP-RC207	5660.12 (69.08)	**5283.58 (34.73)**	5512.13 (41.16)
VRP-RC103	16688.23 (502.44)	**14393.77 (283.84)**	15841.86 (37.09)
VRP-R101	25823.02 (551.51)	**21695.33 (253.92)**	24502.68 (471.45)
VRP-R1_10_1	244992.11 (4287.55)	**184949.94 (2371.5)**	238701.77 (3034.91)
VRP-RC2_10_1	111955.36 (2367.44)	**73007.29 (1911.51)**	100931.69 (2736.13)
VRP-RC1_10_5	222961.69 (2928.4)	**170396.88 (1681.64)**	211305.28 (3506.5)
VRP-C1_10_1	361489.34 (5239.53)	**229452.75 (10321.13)**	353573.25 (5041.99)
VRP-C1_10_8	315320.2 (4759.64)	**207166.83 (4454.09)**	309303.99 (5272.82)
TSP-rat783	**9083.56 (8.03)**	9142.91 (17.01)	9184.83 (15.0)
TSP-usa13509	**23362208.03 (511453.42)**	23891901.99 (1358305.32)	25053111.68 (151623.85)
TSP-u2152	72642.16 (554.91)	72978.04 (813.04)	78817.99 (1121.79)
TSP-pr299	**48640.43 (95.54)**	49266.58 (217.65)	49950.87 (321.47)
TSP-rat575	**6954.3 (7.83)**	6995.99 (12.4)	7023.34 (14.92)
TSP-pcb1173	**59157.25 (106.0)**	59771.29 (214.07)	61332.71 (205.04)
TSP-pr439	115080.16 (1700.91)	**113408.47 (1154.81)**	135922.79 (3307.8)
TSP-d1291	62049.33 (1272.44)	61478.9 (1415.23)	62119.02 (1515.55)
TSP-d18512	684357.59 (2142.31)	704437.29 (31325.38)	792899.46 (3327.09)
TSP-u724	**42914.55 (79.12)**	43290.67 (118.07)	43667.19 (119.84)



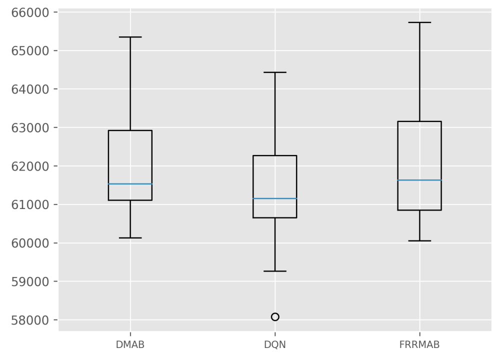
(a) VRP-R201



(b) VRP-C1_10_8



(c) TSP-pr439



(d) TSP-d1291

Figure 2: Boxplot distributions of the objective values achieved by each approach. The smaller the box, the better (less variance)

The main goal of using DQN instead of MAB is to take advantage of a Markovian state representation. In this work, we used the reward information to define the states. In future works, we want to expand the study to use different metrics for representing the search state, like those from Fitness Landscape Analysis. Moreover, with a well-defined MDP, we can investigate the use of other novel reinforcement learning strategies, such as policy optimization and actor-critic models.

ACKNOWLEDGMENTS

This work was funded by the Brazilian agencies CAPES and CNPq.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.

[2] Christian Blum, Jakob Puchinger, Günther R. Raidl, and Andrea Roli. 2011. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing* 11, 6 (2011), 4135 – 4151.

[3] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. 2010. *A Classification of Hyper-heuristic Approaches*. Springer US, Boston, MA, 449–468. https://doi.org/10.1007/978-1-4419-1665-5_15

[4] Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michèle Sebag. 2008. Adaptive Operator Selection with Dynamic Multi-Armed Bandits. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*. Association for Computing Machinery, New York, NY, USA, 913–920. https://doi.org/10.1145/1389095.1389272

[5] Carolina P. de Almeida, Richard A. Gonçalves, Sandra M. Venske, Ricardo Lüders, and Myriam Regattieri Delgado. 2018. Multi-armed Bandit Based Hyper-Heuristics for the Permutation Flow Shop Problem. In *7th Brazilian Conference on Intelligent Systems, BRACIS 2018, São Paulo, Brazil, October 22-25, 2018*. IEEE Computer Society, 139–144. https://doi.org/10.1109/BRACIS.2018.00032

[6] John H. Drake, Ahmed Kheiri, Ender Özcan, and Edmund K. Burke. 2020. Recent advances in selection hyper-heuristics. *European Journal of Operational Research* 285, 2 (2020), 405–428. https://doi.org/10.1016/j.ejor.2019.07.073

[7] A. S. Ferreira, R. A. Gonçalves, and A. Pozo. 2017. A Multi-Armed Bandit selection strategy for Hyper-heuristics. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 525–532. https://doi.org/10.1109/CEC.2017.7969356

[8] Álvaro Fialho. 2010. *Adaptive Operator Selection for Optimization*. Theses. Université Paris Sud - Paris XI. https://tel.archives-ouvertes.fr/tel-00578431

[9] Stephanus Daniel Handoko, Duc Thien Nguyen, Zhi Yuan, and Hoong Chuin Lau. 2014. Reinforcement Learning for Adaptive Operator Selection in Memetic Search Applied to Quadratic Assignment Problem. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO Comp '14)*. Association for Computing Machinery, New York, NY, USA, 193–194.

[10] K. Li, Á. Fialho, S. Kwong, and Q. Zhang. 2014. Adaptive Operator Selection With Bandits for a Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 18, 1 (2014), 114–130. https://doi.org/10.1109/TEVC.2013.2239648

[11] G. Ochoa, M. Hyde, T. Curtois, J.A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A.J. Parkes, S. Petrovic, and E.K. Burke. 2012. HyFlex: A Benchmark Framework for Cross-domain Heuristic Search. 7245 (2012), 136–147.

[12] Martin L. Puterman. 1990. Chapter 8 Markov Decision Processes. In *Handbooks in Operations Research and Management Science*. Stochastic Models, Vol. 2. Elsevier, 331–434.

[13] Jorge A. Soria-Alcaraz, Gabriela Ochoa, Marco A. Sotelo-Figeroa, and Edmund K. Burke. 2017. A methodology for determining an effective subset of heuristics in selection hyper-heuristics. *European Journal of Operational Research* 260, 3 (2017), 972–983. https://doi.org/10.1016/j.ejor.2017.01.042

[14] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning, Second Edition: An Introduction*. MIT Press.

[15] Teck-Hou Teng, Stephanus Daniel Handoko, and Hoong Chuin Lau. 2016. Self-Organizing Neural Network for Adaptive Operator Selection in Evolutionary Search. In *Learning and Intelligent Optimization (Lecture Notes in Computer Science)*, Paola Festa, Meinolf Sellmann, and Joaquin Vanschoren (Eds.). Springer International Publishing, Cham, 187–202.

[16] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-Learning. *Machine Learning* 8, 3 (May 1992), 279–292.

[17] D. H. Wolpert and W. G. Macready. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (April 1997), 67–82.