

A Study on Multi-Objective Optimization of Epistatic Binary Problems Using Q-learning

Yudai Tagawa^a, Hernán Aguirre^b and Kiyoshi Tanaka

Department of Electrical and Computer Engineering, Shinshu University, Wakasato, Nagano, Japan

Keywords: Multi-Objective Optimization, Reinforcement Learning, Q-learning, MNK-landscapes.

Abstract: In this paper, we study distributed and centralized approaches of Q-learning for multi-objective optimization of binary problems and investigate their characteristics and performance on complex epistatic problems using MNK-landscapes. In the distributed approach an agent receives its reward optimizing one of the objective functions and collaborates with others to generate Pareto non-dominated solutions. In the centralized approach the agent receives its reward based on Pareto dominance optimizing simultaneously all objective functions. We encode a solution as part of a state and investigate two types of actions as one-bit mutation operators, two methods to generate an episode's initial state and the number of steps an agent is allowed to explore without improving. We also compare with some evolutionary multi-objective optimizers showing that Q-learning based approaches scale up better as we increase the number of objectives on problems with large epistasis.

1 INTRODUCTION


Multi-Objective Evolutionary Algorithms (MOEAs) (Deb, 2001) have been widely applied to solve real world multi-objective optimization problems, and various types of algorithms have been proposed. MOEAs require further improvements in order to perform an efficient optimization at limited computational cost and cope with problems of increased difficulty, such as large-scale search spaces, many objective functions, and various shapes of the Pareto optimal front set.


In this work we focus on epistatic problems, where the performance of multi-objective optimizers using conventional mutation and recombination operators drops considerably as we increase the number of interacting variables. There is the expectation that in these problems operators guided by learning could lead to improvements. From this standpoint, we study multi-objective optimization using Q-learning (Drugan, 2019) (Watkins and Dayan, 1992), a type of reinforcement learning (RL) (Sutton and Brato, 1998). We want to understand whether Q-learning based search methods perform an effective exploration of large spaces in the presence of epistasis, aiming to develop robust and scalable multi-

objective optimization algorithms.

Related works fall broadly in two categories. Namely, multi-objective reinforcement learning (MORL) and multi-objective optimization combined with reinforcement learning (MOO-RL). The emphasis of MORL is the multi-objective sequential decision making of the agents to learn to perform a task when the reward space is multi-dimensional. Several MORL algorithms have been proposed. Most of them use linear scalarization functions to map the reward vector into a scalar (Lizotte et al., 2010) (Gábor et al., 1998) (Barrett and Narayanan, 2008) (Hayes et al., 2022) (Moffaert et al., 2013b) (Moffaert et al., 2013a).

On the other hand, MOO-RL emphasises multi-objective solution search supported by RL, i.e. blending multi-objective optimizers with RL. MOO-RL can be subdivided in two major categories. One where the solution search is carried out by the optimizer applying its operators of variation and selection whereas RL is applied to select strategies or configurations for the optimizer. There are a few works in this direction, for example, Q-learning has been used in dynamic multi-objective optimization to select global and local search strategies to be applied by a memetic algorithm (Shen et al., 2018) and to select strategies to initialize the population of the multi-objective optimizer (Zou et al., 2021) every time a critical dynamic event occurs.

^a  <https://orcid.org/0009-0005-4370-7633>

^b  <https://orcid.org/0000-0003-4480-1339>

The other major category for MOO-RL is where RL is used as a multi-objective optimizer. That is, a state includes the codification of a solution to the optimization problem and actions act as operators of variation to search in the solution space. There are very few previous works on RL applied as a multi-objective optimizer. For example, in (Mariano and Morales, 2000) a distributed approach was used to optimize 2 and 3 objective functions with two continuous variables. In (Jalalimanesh et al., 2017), a distributed Q-learning algorithm similar to (Mariano and Morales, 2000) is applied for multi-objective optimization of radiotherapy aiming to find Pareto-optimal solutions representing radiotherapy treatment plans.

We focus on the latter category of MOO-RL and study distributed and centralized approaches of Q-learning for multi-objective optimization of binary problems. In the distributed approach an agent receives its reward optimizing one of the objective functions and collaborates with others to generate Pareto non-dominated solutions. In the centralized approach the agent receives its reward based on Pareto dominance optimizing simultaneously all objective functions.

In order to understand the characteristics of the RL approaches, we conduct experiments solving MNK-landscapes (Aguirre and Tanaka, 2007) varying the number of binary variables N , the number of objectives M and the number of interacting variables K (epistatic interactions). We compare results with other MOEAs using 100 bits landscapes. We chose for the comparison the multi-objective random bit climber moRBC (Aguirre and Tanaka, 2005), the NSGA-II (Deb et al., 2002) and the decomposition based MOEA/D (Zhang and Li, 2008) algorithms, which performance is known on MNK-landscapes and thus allow us to better understand the effectiveness of the actions and reward approaches of the RL optimizers on terms of well known selection methods and operators of variation as we scale up objective space and epistatic interactions between variables. We show that Q-learning based approaches can perform significantly better than the other algorithms on increasingly non-linear problems for a broad range of K . We also show that the comparison with the other algorithms provides valuable insights on how to further improve Q-learning approaches for epistatic problems.

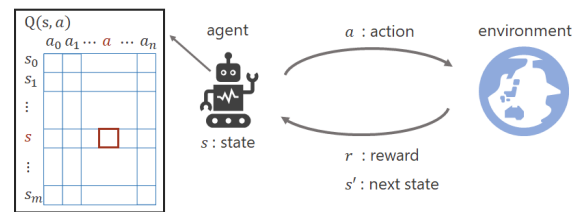


Figure 1: Q-learning.

2 METHOD

2.1 Q-learning

Reinforcement learning (RL) is a method in which an agent learns what to do in given situations so as to maximize a numerical reward signal. In RL an agent is not told which actions to take, but instead must discover which actions yield the most reward by trying them. Q-learning is a type of RL that uses an off-policy temporal difference control algorithm to learn an action-value function Q , which approximates the optimal action-value function independently of the policy being followed (Watkins and Dayan, 1992). Fig. 1 illustrates the main components of Q-learning. When an agent takes action a in state s , a reward r and the next state s' are passed from the environment. The value of Q is updated by the following equation,

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

where α is the learning rate and γ is the discount rate, a constant between 0 and 1. The above updating equation means that when an action causes a transition from the current state s to the next state s' , its Q-value is brought closer to the value of the action a' with the highest Q-value in the next state s' . This means that if a state has a high reward, that reward will propagate to the states that can reach that state with each update. This results in optimal learning of state transitions. The interaction between the agent and the environment is repeated until a terminal state has been reached. Each time an interaction takes place is called a *step* and an *episode* denotes the multiple steps of interaction taken from the initial state to the terminal state. Distributed Q-learning (Mariano and Morales, 2000) is a method where multiple agents interact with the environment while usually referring to the same Q-table.

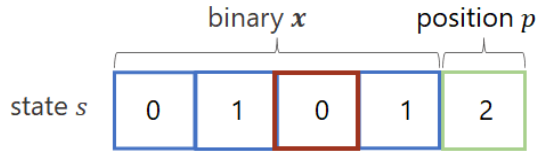


Figure 2: State.

2.2 Q-learning for Multi-Objective Optimization

2.2.1 Main Components

To apply Q-learning, the environment, states, actions and rewards have to be properly defined. In this work we focus on the optimization of multi-objective binary problems and the Q-learning components are defined to reflect that. Let us denote the vector x of binary variables as the environment where an agent can move and allow an agent at a given time to be positioned in one of the variables of the vector. Hence, a state s is represented by joining the binary solution instantiated in $x = (x_0, \dots, x_{n-1})$ and the position p of the agent, where $x_i \in \{0, 1\}$, n denotes the number of variables and $p \in \{0, \dots, n-1\}$. The total number of states is $n \times 2^n$ with this representation. Fig. 2 illustrates the representation of a state s for $n = 4$ variables. An action causes the agent to move to another variable and flip its value. In other words, an action serves as a variation operator that mutates one bit of a solution to create a new one. We investigate two kinds of actions to transition from one state to another, which are detailed later in this section together with the way the reward is assigned.

We study a distributed and a centralized approach for solving the task of multi-objective optimization. In the distributed approach an agent focuses on a particular objective function of the problem and its actions are rewarded for its relative quality in that objective function. Thus, multiple agents are required to cooperate, at least one agent per objective function, to solve the task in the distributed approach. In the centralized approach, an agent focuses on all objective functions and its actions are rewarded for its relative quality in the multi-objective space. In the following, for short, we refer to the agents used in the distributed approach as single-objective agents and to the agents used in the centralized approach as multi-objective agents. In both cases the objective is to find a set of Pareto solutions.

2.2.2 Multi-Agent Algorithm Framework

We implement a tunable multi-agent algorithm framework to investigate multi-objective optimization using

Algorithm 1: Multi-objective optimization framework using Q-learning.

Data: $init_type, agent_type, act_type, E, M, \tau$
Result: P , the set of non-dominated solutions found by the algorithm

```

1  $Q \leftarrow \text{Initialize}Q()$ 
2  $P \leftarrow \{\}$ 
3 for 1 to  $E$  do /* episodes */
4    $S \leftarrow \{\}$ 
5   for  $i \leftarrow 1$  to  $M$  do /* agents */
6      $s \leftarrow \text{InitializeState}(init\_type, P)$ 
7      $x \leftarrow \text{GetSolution}(s)$ 
8      $P_i \leftarrow \{x\}$ 
9      $c \leftarrow 0$ 
10    while  $c \leq \tau$  do
11      /*  $i$ -th agent steps */
12       $a \leftarrow \text{SelectAction}(act\_type, s, Q)$ 
13       $s' \leftarrow \text{PerformAction}(s, a)$ 
14       $x \leftarrow \text{GetSolution}(s')$ 
15       $r \leftarrow \text{ObserveReward}(agent\_type, x, P_i)$ 
16       $S \leftarrow S + (s, a, s', r)$ 
17       $P_i \leftarrow P_i \cup \{x\}$ 
18       $s \leftarrow s'$ 
19       $c \leftarrow \text{UpdateCounter}(agent\_type, x, P_i)$ 
20    end
21  foreach  $(s, a, s', r)$  in  $S$  do
22     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} (Q(s', a')) - Q(s, a)]$ 
23  end
24   $P \leftarrow \text{NonDominatedSolutions}(P \bigcup_{i=1, \dots, M} P_i)$ 
25 end
26 return  $P$ 

```

Q-learning either in a distributed or centralized approach. Algorithm 1 illustrates the framework. In the following we explain relevant details of the algorithm.

First, the quality table Q is initialized to zero for each combination of state-action, i.e. $\forall s \wedge \forall a Q(s, a) = 0.0$, and the bounded population P of non-dominated solutions is set to empty (lines 1-2). Next, the algorithm iterates for E episodes for each of the M specified agents (lines 3-24) and returns the set of non-dominated solutions found (line 26). In this work, if P exceeds its specified size it is truncated using crowding distance (Deb, 2001) (line 24).

The information associated to a step taken by an agent is given by the tuple (a, s, s', r) , where a is the action, s is the current state, s' is the next state and r is

the reward. When an episode starts, the list S that will contain the information of all the steps taken by all agents during an episode is initialized to empty (line 4). Before the first step of an episode for the i -th agent is taken, an initial state is defined, the population P_i of solutions visited by the agent during an episode is initialized with the solution x contained in the initial state s , and the counter c used to verify the termination of an episode is set to 0 (lines 6-9).

In each step of an episode, an action is selected and executed so that the i -th agent transitions from the current state s to a new state s' . The solution x contained in s' is compared with the population of solutions P_i collected so far by the agent to compute the reward of the action, the tuple (a, s, s', r) is added to S , the solution x is added to P_i , and the new state s' becomes the current state s (lines 11-18).

Once all M agents have completed an episode, the quality table Q is updated with the information collected in S of all the steps taken by all agents during the episode (line 21-23), and the set of non-dominated solutions is updated with the solutions visited by the agents contained in their respective populations P_i (lines 24).

2.2.3 Initial State

Two methods for generating an initial state (line 6) at the start of an agent's episode are studied. One of the methods generates randomly the solution x associated to the initial state and the other one chooses a solution x from the set of non-dominated solutions P_i collected so far. In both methods, the position p of the agent is randomly determined. We select between these methods setting *init_type* either to *randomly* or *continuously*, respectively.

2.2.4 Types of Actions and Solution Generation

Two types of actions called right-left (*rl*) and *anywhere* are studied. The *rl* action moves the agents from its current position p to either the right $p + 1$ or left $p - 1$ neighboring positions. On the other hand, the *anywhere* action moves the agent from its current position p to any of the n positions in the vector x , including p again. In both kinds of actions, the bit in the position where the agent moves is flipped.

Fig. 3a shows an example of moving to the right from the current position $p = 1$, when the type of action is *rl*. The position after the move is $p' = 2$, and the next state s' is formed by joining x' and p' . We consider the vector x as a circular array. That is, the position to the right of $p = n - 1$ is $p' = 0$. Similarly, the position to the left of $p = 0$ is $p' = n - 1$. When the type of action is *rl*, the number of the ac-

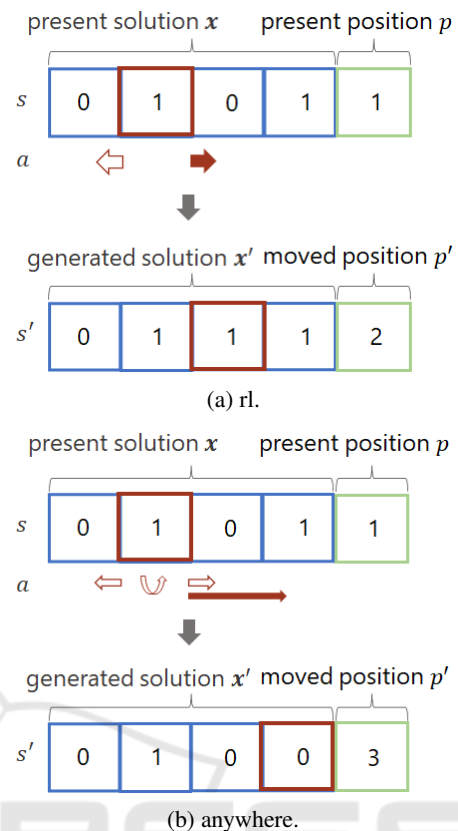


Figure 3: Types of actions.

tions an agent can choose from is 2, either right or left, independently of the dimension n of the vector x . Fig. 3b shows an example of moving from the current position $p = 1$ to $p' = 3$ when the type of action is *anywhere*. Since this example is a 4-bit problem, the number of actions an agent can choose from is 4. In general, when the type of action is *anywhere*, the number of actions an agent can choose from is n , the dimension of the vector x . In the framework, we select between these two types of actions by setting *act_type* to either *rl* or *anywhere*. In this work, the agents select probabilistically the action in the current state using an ϵ -greedy strategy. That is, with probability $1 - \epsilon$ the action with the highest Q-value in the current state s is chosen, and with probability ϵ the action is chosen randomly.

2.2.5 Reward Assignment

Rewards are given in different ways depending on the type of agent. In the case of distributed agents, if the generated solution x improves the fitness value of the best solution in P_i , in the fitness function the agent is in charge of, the agent receives a positive reward equal to the size of P_i . Otherwise, the reward is negative and equal to the number of solutions in P_i that are better

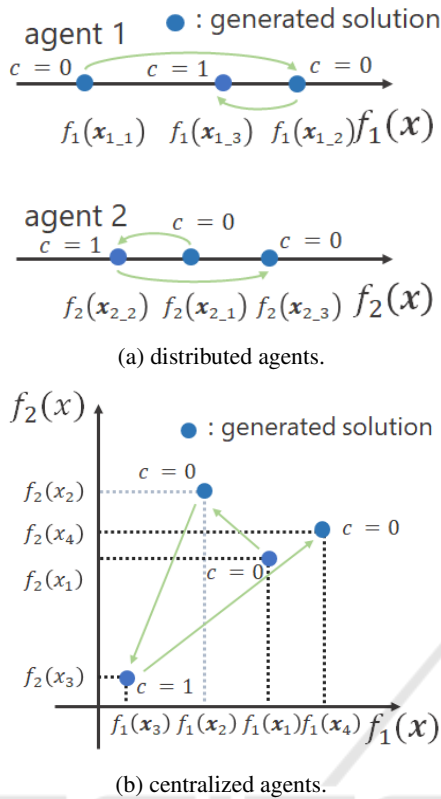


Figure 4: Distributed and centralized agents behavior.

than x . In the case of centralized agents, the generated solution x is compared using the Pareto dominance relationship with the non-dominated solutions in P_i . If x is dominant, the agent receives a positive reward equal to the number of solutions that x dominates. If x is dominated, the reward is negative and equal to the number of solutions that dominate x . Otherwise, if x is nondominated by P_i , the reward is -1 .

2.2.6 Agent's Episode Termination Condition

To determine whether an agent has reached a terminal state (line 10) we keep a counter c of the number of consecutive times an agent i fails to improve the best solutions in its corresponding population P_i . Once this counter goes above a threshold τ , $c > \tau$, the episode for that agent ends. In the case of distributed single-objective agents, the counter c increases if the fitness value of the solution x extracted from the new state (line 13), in the corresponding fitness function the i -th agent is assigned to, does not improve the fitness value of the best solution in P_i . In the case of centralized multi-objective agents, the counter c increases if solution x extracted from the new state (line 13) is Pareto dominated by at least one solution in P_i . Fig. 4a and 4b illustrate the single-objective and multi-objective agents search and how the counter c is

Table 1: Parameters: MNK-landscapes.

parameter	MNK-landscapes
Objectives M	2, 3, 4
Variables N	100
Interacting Variables K	1, 2, 3, 5, 7, 10, 15, 20
Variables Interaction	random

Table 2: Parameters: Q-learning.

parameter	Q-learning
Episodes	$\leq 2 \times 10^6$ evaluations
Agent Type	<i>single, multi</i>
Action Type	<i>rl, anywhere</i>
Initial State	<i>continuously</i>
τ	0
ϵ, α, γ	0.1, 0.1, 0.6
Population size	100

updated when they optimize a two objective problem.

3 EXPERIMENTS

We compare the performance of Q-learning based multi-objective optimization with NSGA-II (Deb et al., 2002), the multi-objective random bit climber moRBC (Aguirre and Tanaka, 2005) and MOEA/D (Zhang and Li, 2008) using large MNK-landscapes with $M = 2, 3$ and 4 objectives, $N = 100$ bits, varying the number of epistatic bits K from 1 to 20. In these experiments all algorithms run until 200,000 fitness evaluations have been completed. Parameters of the MNK-landscapes used in our study are summarized in Table 1. Parameters used for Q-learning are summarized in Table 2 and parameters for the other MOEAs in Table 3.

In all experiments, results are reported for 10 trials of the algorithms in the same MNK-landscape with different random seeds. We use Hypervolume (HV) (Zitzler, 1999) as the evaluation metric setting the reference point to $(0, \dots, 0)$.

4 RESULTS AND DISCUSSION

In this section we observe the performance of the centralized and distributed approaches using the two different types of action, varying the number of objectives M from 2 to 4 and the number of interacting bits K from 1 to 20. This allows us to understand better the Q-learning based approaches when we scale up the dimension of the objective space and the complexity of the landscape. In the following experiments

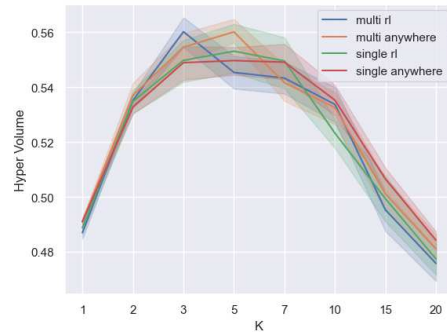
Table 3: Parameters: Other MOEAs.

parameter	NSGA-II	moRBC	MOEA/D
Generations	2000	2000	2000
Population size	100	100	100
Crossover	two-point	-	two-point
Mutation	bit flip	bit flip	bit flip
Neighborhood size	-	-	20
Scalarized function	-	-	Tchebycheff

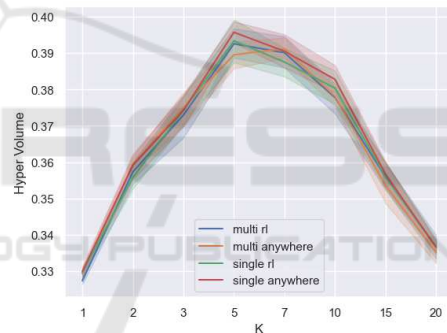
we fix τ to 0, the threshold for the counter of the number of consecutive times an agent fails to improve the best solutions in its corresponding population. This threshold has shown best results in our experiments. Also we use a solution selected from the population P of non-dominated solutions to generate the initial state of an episode, i.e. *continuously* strategy.

Fig. 5 plots HV over K for all four possible combinations agent type and action type. Results show the HV of the final population P of nondominated solutions after 200,000 fitness evaluations. Note that for 2 objectives, the multi-objective agent perform better when K is low, and the single-objective agent with *anywhere* action performed better when K is high. For 3 and 4 objectives, the single-objective agents with *anywhere* action achieves the highest HV. Note that the centralized approach with a multi-objective agent and *anywhere* action can perform better than the distributed approaches only for $M = 2$ objectives and $2 \leq K \leq 5$. In all other cases, $M = 2$ for $K \geq 7$ and $M = 3, 4$ for all values of K , the distributed approach with single-objective agents and *anywhere* action overall performed better. As the dimension of the objective space increases it becomes clear that the centralized approach using a reward given by Pareto dominance does not scale up well, as seen in Fig. 5c for $M = 4$. A centralized approach offers the possibility to reduce the number of agents required for the multi-objective search. However, results in this work clearly suggest that a reward based on Pareto dominance could only be effective in a very limited subset of problems. It could be worth exploring in the future other forms to assign rewards for a centralized agent. The *rl* action overall does appear superior to *anywhere* in terms of performance. However, the combined states-action space by *rl* is significantly smaller than by *anywhere*. Actions *rl* and *anywhere* can be seen as extreme cases in terms of the size of the neighborhood of the position codified in the state where a bit can be mutated. It could be useful to explore actions where the size of the current position's neighborhood is between 2 (*rl*) and n (*anywhere*).

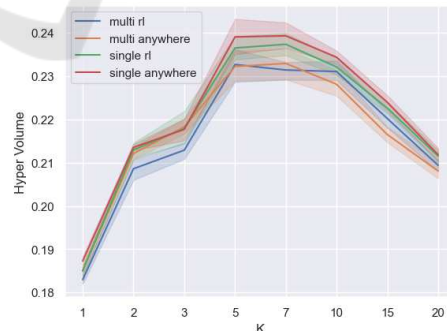
Next, we compare the Q-learning distributed approach using action *anywhere* for multi-objective op-



(a) M2N100.



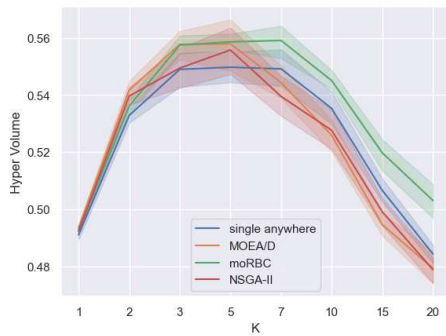
(b) M3N100.



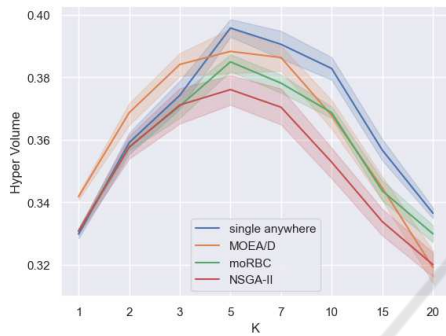
(c) M4N100.

Figure 5: Agent Types and Action Types (100-bits).

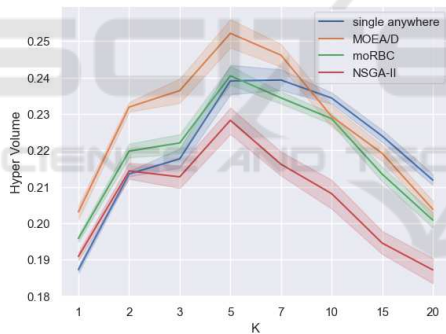
timization with NSGA-II, moRBC and MOEA/D running for the same number of fitness evaluations as the Q-learning based approaches (200,000) setting their population to 100. Fig. 6 shows HV over K , similar to Fig. 5.



(a) M2N100.



(b) M3N100.



(c) M4N100.

Figure 6: Comparison to NSGA-II, moRBC and MOEA/D.

Before we discuss in detail this figure it is worth remembering some properties of MNK-landscapes. By enumeration it has been shown that increasing $K > 0$ the landscape becomes rugged and the peaks' height increase until medium values of K . Thereafter the peaks remain of similar height to large values of K . The hypervolume of the true Pareto front follows a trend similar to the height of the peaks (Aguirre and Tanaka, 2007).

Now, looking at the Fig. 6, it should be noted that the increase in hypervolume varying K from 1 to 5 for all algorithms is in accordance with the properties of the landscapes. However, for $K \geq 7$ the hypervolume decreases monotonically with K for all algorithms, which means that the performance of all al-

gorithms drops substantially for $K \geq 7$. Also, note that there is not a dominant algorithm for all K and M . However some important trends can be observed. The Q-learning based approach is the best performing algorithm in 3 and 4 objectives for $K \geq 5$ and $K \geq 10$, respectively, and the second best for 2 objectives and $K \geq 7$. It is also notoriously weak in all objectives for $K \leq 3$. On the other hand, MOEA/D is a very strong algorithm in 2, 3 and 4 objectives for $K \leq 5$, but its performance drops faster than moRBC and the Q-learning approach for $K \geq 7$. The moRBC is overall the strongest algorithm in 2 objectives for $K \geq 3$ and similar or better than NSGA-II for all K and M . NSGA-II is competitive only in 2 objectives for $K \leq 2$ and scales up badly for 3 and 4 objectives for all K .

The difference in performance among algorithms is due to the combined effectiveness of the operators of variation and selection included in the algorithms. The Q-learning based approach, moRBC and NSGA-II use Pareto dominance based ranking in their selection mechanism. It is well known that increasing the dimension of the objective space algorithms with this kind of ranking scale up poorly, compared to a decomposition based approach like MOEA/D. In addition, in smooth landscapes the regions of non-dominance are broad and solutions in the Pareto front are evenly distributed. Thus, it is not surprising that MOEA/D with its uniform distribution of weights outperforms the other algorithms for small K . However, as K increases and the landscapes become rugged the regions of non-dominance become fragmented and smaller, inducing not uniform Pareto fronts where solutions are more separated in objective and decision space (Aguirre and Tanaka, 2007). Here the effectiveness of the operators of variation becomes more relevant, in addition to selection. MOEA/D for large K keeps the relative advantage of its selection mechanism for 3 and 4 objectives, but the combination of crossover and mutation loses effectiveness. The better performance by moRBC compared with NSGA-II is explained by the thorough exploration of local optima by one-bit mutations rather than by more disruptive operators like crossover. The actions in the Q-learning approach are also one-bit mutations. The Q-table offers a path to improving moves once an episode is restarted, guiding the exploitation of promising regions and climbing to better local optima, which becomes more difficult without learning as evidenced by the results for large K . However, different to moRBC, the actions in the Q-learning approach allow transitions to states with non-improving solutions and are far less comprehensive to explore local optima.

The results by all algorithms compared in this section provide valuable insights to improve the Q-learning approach. They suggest that incorporating some of the functionality of the moRBC climber into the transitions allowed for the Q-learning approach could improve its effectiveness. In addition, ways to include selection principles that are more robust in objective spaces of larger dimensions should be considered. This implies different ways to compute the rewards and the selection of the solution to restart an episode.

5 CONCLUSION

In this work, we studied distributed and centralized approaches of Q-learning for multi-objective optimization of binary epistatic problems using MNK-landscapes. We showed that the Q-learning based approaches scale up better than moRBC, NSGA-II and MOEA/D as we increase the number of objectives on problems with large epistasis. Also, we identified their weaknesses particularly in low epistatic landscapes. In addition, we analyzed results of other MOEAs taking into account their selection method and operators of variation together with properties of MNK-landscapes to better understand the Q-learning based approaches and suggested forms to improve them. Our conclusions regarding the parameters of the Q-learning based approaches are as follows. The action that flips any bit is overall slightly superior to the action that flips the left or right neighboring bits. The centralized approach, using a reward based on Pareto dominance, does not scale up well with the dimension of the objective space.

In the future, we would like to study other ways to assign rewards for the centralized approach, enhance the selection of solutions for the initial state of an episode, and constraint transitions to non-improving states. We would also like to study the Q-learning approaches for many-objective optimization and analyze the optimization history obtained by Q-learning.

REFERENCES

- Aguirre, H. and Tanaka, K. (2005). Random Bit Climbers on Multiobjective MNK-Landscapes: Effects of Memory and Population Climbing. *IEICE Transactions*, 88-A:334–345.
- Aguirre, H. and Tanaka, K. (2007). Working Principles, Behavior, and Performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, 181:1670–1690.
- Barrett, L. and Narayanan, S. (2008). Learning All Optimal Policies with Multiple Criteria. *International Conference on International Conference on Machine Learning*, pages 41–47.
- Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Drugan, M. (2019). Reinforcement Learning Versus Evolutionary Computation: A Survey on Hybrid Algorithms. *Swarm Evol. Comput.*, 44:228–246.
- Gábor, Z., Kalmár, Z., and Szepesvári, C. (1998). Multi-Criteria Reinforcement Learning. *International Conference on International Conference on Machine Learning*, 98:197–205.
- Hayes, C., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., and et al (2022). A Practical Guide to Multi-Objective Reinforcement Learning and Planning. *Autonomous Agents and Multi-Agent Systems*, 32(1):26.
- Jalalimanesh, A., Haghighi, H. S., Ahmadi, A., Hejazian, H., and Soltani, M. (2017). Multi-Objective Optimization of Radiotherapy: Distributed Q-Learning and Agent-Based Simulation. *Journal of Experimental & Theoretical Artificial Intelligence*, 29(5):1071–86.
- Lizotte, D., Bowling, M., and Murphy, S. (2010). Efficient Reinforcement Learning with Multiple Reward Functions for Randomized Controlled Trial Analysis. *International Conference on International Conference on Machine Learning (ICML)*, 10:695–702.
- Mariano, C. and Morales, E. (2000). Distributed Reinforcement Learning for Multiple Objective Optimization Problems. In *Proc. of Congress on Evolutionary Computation (CEC-2000)*, pages 188–195.
- Moffaert, K. V., Drugan, M., and Nowé, A. (2013a). Hypervolume-Based Multi-Objective Reinforcement Learning. *Evolutionary Multi-Criterion Optimization*, pages 352–66.
- Moffaert, K. V., Drugan, M., and Nowé, A. (2013b). Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–99.
- Shen, X., Minku, L., Marturi, N., Guo, Y., and Han, Y. (2018). A Q-Learning-Based Memetic Algorithm for Multi-Objective Dynamic Software Project Scheduling. *Information Sciences*, 428:1–29.
- Sutton, R. and Brato, A. (1998). *Reinforcement Learning*. The MIT Press.
- Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.
- Zhang, Q. and Li, H. (2008). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Neural Netw.*, 11(6):712–731.
- Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland.

Zou, F., Yen, G., Tang, L., and Wang, C. (2021).
A Reinforcement Learning Approach for Dynamic
Multi-Objective Optimization. *Information Sciences*,
546:815–34.

