Scientific Research Publishing

Open Journal of Optimization, 2017, 6, 65-84
http://www.scirp.org/journal/ojop
ISSN Online: 2325-7091
ISSN Print: 2325-7105

A Novel Approach Based on Reinforcement Learning for Finding Global Optimum

Cenk Ozan[1], **Ozgur Baskan**[2], **Soner Haldenbilen**[2]

[1]Department of Civil Engineering, Faculty of Engineering, Adnan Menderes University, Aydin, Turkey
[2]Department of Civil Engineering, Faculty of Engineering, Pamukkale University, Denizli, Turkey
Email: *cenk.ozan@adu.edu.tr

How to cite this paper: Ozan, C., Baskan, O. and Haldenbilen, S. (2017) A Novel Approach Based on Reinforcement Learning for Finding Global Optimum. *Open Journal of Optimization*, **6**, 65-84.
https://doi.org/10.4236/ojop.2017.62006

Received: March 31, 2017
Accepted: June 25, 2017
Published: June 28, 2017

Abstract

A novel approach to optimizing any given mathematical function, called the **MO**dified **RE**inforcement **L**earning **A**lgorithm (MORELA), is proposed. Although Reinforcement Learning (RL) is primarily developed for solving Markov decision problems, it can be used with some improvements to optimize mathematical functions. At the core of MORELA, a sub-environment is generated around the best solution found in the feasible solution space and compared with the original environment. Thus, MORELA makes it possible to discover global optimum for a mathematical function because it is sought around the best solution achieved in the previous learning episode using the sub-environment. The performance of MORELA has been tested with the results obtained from other optimization methods described in the literature. Results exposed that MORELA improved the performance of RL and performed better than many of the optimization methods to which it was compared in terms of the robustness measures adopted.

Keywords

Reinforcement Learning, Mathematical Function, Global Optimum, Sub-Environment, Robustness Measures

1. Introduction

If $f(x)$ is a function of decision variables, where $x \in S$, S is the feasible search space and $S \subseteq R^n$, an optimization problem can be defined as finding the value of x_{best} in S that makes $f(x)$ optimal for all x values. Despite the fact that different meta-heuristic algorithms have been improved especially in last two decades, the contributions of Reinforcement Learning (RL) to this area are still limited comparing to others. Numerous studies such as genetic algorithm based methods [1] [2], ant colony based algorithms [3] [4], harmony search approach [5], modified firefly algorithm [6] and hybrid heuristic methods [7] [8] [9] have been proposed to optimize any given mathematical function. Apart from these applications, there are some attempts for solving engineering problems in different fields using RL based algorithms in the relevant literature. Hsieh and Su [10] developed Q-learning based optimization algorithm for solving economic dispatch problem after they tested their proposed algorithm for optimizing standard mathematical test functions. Their results showed that the proposed algorithm outperformed many existing optimization algorithms. Similarly, Samma *et al.* [11] tested newly developed RL based Memetic Particle Swarm Optimization algorithm on mathematical functions and real-world benchmark problems. Numerical examples applied indicated that the proposed algorithm is able to produce better results than those did by other optimization algorithms given in the literature. Walraven *et al.* [12] proposed a new algorithm to minimize traffic flow using RL algorithm. They formulated traffic flow optimization problem as Markov Decision Process and used Q-learning method to reduce traffic congestion. Another application of the RL algorithm has been proposed by Tozer *et al.* [13]. They developed a new RL algorithm that is able to find optimal solution with several conflicting objectives. The proposed algorithm has been tested on multi-objectives path finding problems with deterministic and stochastic environments.

Although most of the methods described in the literature are able to discover global optimum of any given optimization problem, the performance of newly developed algorithms should be investigated. Therefore, we present a **MO**dified **RE**inforcement **L**earning **A**lgorithm (MORELA) approach which differs from RL based approaches by means of generating a sub-environment based on the best solution obtained so far which is saved to prevent the search being trapped at local optimums. And then, all of the function values with corresponding decision variables in MORELA are ranked from best to worst. In this way, the sub-environment is compared with the original environment. If one of the members of the sub-environment produces better functional value, it is added to the original environment and the worst solution is omitted. This makes the searching process more effective because the global optimum is sought around the best solution achieved so far, with the assistance of the sub-environment and the original environment.

RL has been attracted a lot of attention from scientific community for solving different class of problems especially in last decades [14]. In RL, there is a remarkable interaction between agent and environment which contains everything apart from the agent. The agent receives information from the environment through cooperating with each other. Depending on the information obtained, the agent changes the environment by means of implementing an action. This modification is transferred to the agent by means of a signal. The environment generates numerical values called rewards and the agent efforts to maximize them. The agent and environment affect each other at each time t in which the agent gets information about the environment's state $s_t \in S$, where S consists of states. To this respect, the agent performs an action $a_t \in A(s_t)$, where $A(s_t)$ includes actions in state s_t and gains a reward $r_{t+1} \in R$, and it is located in a new state s_{t+1}. Figure 1 represents this cooperation [15].

There are three primary types of RL based methods, each with its advantages and disadvantages. The Monte Carlo and Temporal difference learning methods are able to learn only from experience whereas the other one, called Dynamic programming, requires a model of the environment. Therefore, because of not need to have a model, they are superior to Dynamic programming. Indeed, temporal difference learning methods are at the core of RL [16]. On the other hand, *Q*-learning, one of the temporal difference methods, evaluates Q values, which represent quality of a given state-action pair [17]. It benefits experience to update members of Q table [14]. Q table has elements as $Q(s,a)$ for each state-action pair. The *Q*-learning determines the Q value, which reflects an action *a* performed in a state *s*, and selects the best actions [18]. The Q table is created as shown in Table 1 [19].

Q learning algorithm includes in a sequence of learning episodes (*i.e.* iteration). At each learning episode, the agent chooses an action in accordance with information provided from a state *s*. The agent deserves to receive a reward considering its Q value and observes the next state, s'. The agent replaces its Q value with that provided according to Equation (1):

$$Q_t(s,a) \leftarrow (1-\alpha)\times Q_t(s,a)+\alpha\times\left[\overbrace{r_t(s,a)+\gamma\times Q_{t-1}^{best}(s,a)}^{\text{next state } s'}\right] \quad (1)$$

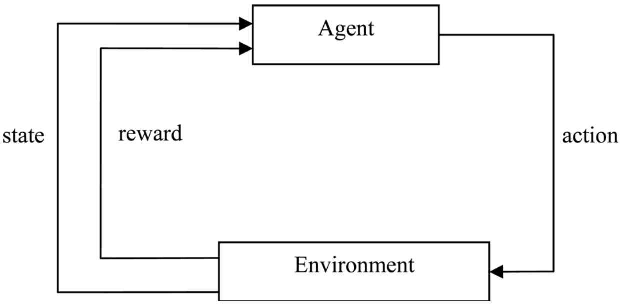


Figure 1. The cooperation between agent and environment.

Table 1. Q learning process.

```
Initialize $Q$ values
Repeat $t$ times ($t$ = number of learning episodes)
    Select a random state $s$
    Repeat until the end of the learning episode
        Select an action $a$
        Receive an immediate reward $r$
        Observe the next state $s'$
        Update the $Q$ table according to the update rule
        Set $s = s'$
```

where $Q_t(s,a)$ is the updated Q value, $Q_{t-1}(s,a)$ is the Q-value saved in the Q table, $r_t(s,a)$ is the reward for state-action pair, α is the learning rate, and γ is the discounting parameter [18].

This paper proposes a new and robust approach, called MORELA, to optimize any given mathematical function. MORELA approach varies from other RL based approaches through generating a sub-environment. In this way, developed MORELA approach has ability to find global optimum for any given mathematical optimization because it is sought both around the best solution achieved so far with the assistance of the sub-environment and the original environment. The rest of this paper is organized as follows. The definition of fundamental principles of MORELA is provided in Section 2. Section 3, which also contains comparison of MORELA and RL, various contrastive analyses of MORELA such as robustness analysis, comparisons with other related methods, explanation of evolving strategy of MORELA and investigation of the effect of high dimensionality, presents numerical experiments and the last section is conclusions.

2. The MORELA Approach

There are several studies related to RL combined with different heuristic methods for solving different types of optimization problems. Liu and Zeng [20] developed genetic algorithm based method with assistance of reinforcement mutation to tackle the problem of travelling salesman. Integrating RL with different algorithms has been used to address the problem of robot control with unrecognized obstacles [21]. The experimental results revealed that the hybrid approach is superior to RL for planning robot motion whereas RL faced some difficulties. Chen *et al.* [22] proposed genetic network programming with RL for stock trading. The comparison of the results with those obtained from other methods shows the noticeable performance of their method. Similarly, Wu *et al.* [23] improved a RL based method considering multi-agent for tackling job scheduling problems. Their results showed that the method is comparable to that of some centralized scheduling algorithms.

From a different viewpoint, Derhami *et al.* [24] applied RL based algorithms for ranking most relevant web pages to user's search. The algorithms are tested by using well-known benchmark datasets. Their results showed that the use of RL makes noticeable improvements in ranking of web pages. Khamis and Gomaa [25] used an adaptive RL approach to tackle traffic signal control at junctions by considering multi-objective. Recently, Ozan *et al.* [26] developed RL based algorithm in determining optimal signal settings for area traffic control. Results revealed that the proposed algorithm outperforms genetic algorithm and hill climbing methods even if there is a heavy demand condition.

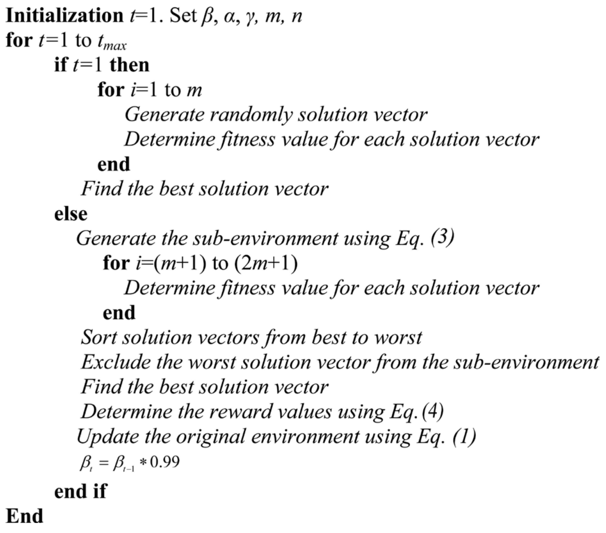
Hybridizing RL algorithms with other optimization methods is a powerful technique to tackle different types of optimization problems arising in different fields. Thus, in the context of this paper, we focus on applicability of RL based algorithms to discover global optimum for any mathematical function. The proposed algorithm called MORELA is on the basis of Q-learning, which is a model-free RL approach. In addition, a sub-environment is generated in MORELA so that the environment consists of original and sub-environment differently from other RL based approaches as shown in Equation (2) [26].

$$\begin{bmatrix} Q_t^{11}(s,a) & Q_t^{12}(s,a) & \cdots & Q_t^{1n}(s,a) \\ Q_t^{21}(s,a) & Q_t^{22}(s,a) & \cdots & Q_t^{2n}(s,a) \\ \vdots & \vdots & & \vdots \\ Q_t^{m1}(s,a) & Q_t^{m2}(s,a) & \cdots & Q_t^{mn}(s,a) \\ Q_{t-1}^{(m+1)1}(s,a) & Q_{t-1}^{(m+1)2}(s,a) & \cdots & Q_{t-1}^{(m+1)n}(s,a) \\ Q_t^{(m+2)1}(s,a) & Q_t^{(m+2)2}(s,a) & \cdots & Q_t^{(m+2)n}(s,a) \\ \vdots & \vdots & & \vdots \\ Q_t^{(2m+1)1}(s,a) & Q_t^{(2m+1)2}(s,a) & \cdots & Q_t^{(2m+1)n}(s,a) \end{bmatrix} \Rightarrow \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \\ \vdots \\ f_{2m+1} \end{bmatrix} \tag{2}$$

where m is the size of the original environment, n is the number of decision variables, and f is fitness value at the t^{th} learning episode. As shown in Equation (2), $Q(s,a)$ value achieved in the previous learning episode is kept in the $(m+1)^{\text{th}}$ row. At the t^{th} learning episode, a sub-environment is generated as given in Equation (3) and located between rows $(m+2)$ and $(2m+1)$. Thus, a global optimum is explored around the best solution with assistance of sub-environment with vector of β which must be decreased during algorithm process in order to make searching more effectively in a reduced environment. The limits of β can be selected by considering upper and lower constraints of a given problem [3].

$$rnd\ \left(Q_{t-1}^{best}(s,a)-\beta; Q_{t-1}^{best}(s,a)+\beta\right) \tag{3}$$

After generating the sub-environment, the solution vectors located in both environments are ranked from best to worst according to their fitness values. With assistance of this sorting, the worst solution vector is excluded from environment whereas the solution vector provided a better functional value is included. Thus, MORELA may gain the ability to solve any given optimization problem without prematurely converging. Figure 2 shows the process of MORELA.



```
Initialization $t$=1. Set $\beta, \alpha, \gamma, m, n$
for $t$=1 to $t_{max}$
      if $t$=1 then
            for $i$=1 to $m$
                  Generate randomly solution vector
                  Determine fitness value for each solution vector
            end
            Find the best solution vector
      else
            Generate the sub-environment using Eq. (3)
            for $i$=($m$+1) to (2$m$+1)
                  Determine fitness value for each solution vector
            end
            Sort solution vectors from best to worst
            Exclude the worst solution vector from the sub-environment
            Find the best solution vector
            Determine the reward values using Eq. (4)
            Update the original environment using Eq. (1)
            $\beta_t = \beta_{t-1} * 0.99$
      end if
End
```

Figure 2. The process of MORELA.

In MORELA, each action a in state s is rewarded as shown in Equation (4) [26].

$$r_t(s,a) = \frac{Q_t^{best}(s,a) - Q_t(s,a)}{Q_t(s,a)} \tag{4}$$

where $r_t(s,a)$ is the reward function, $Q_t(s,a)$ is the Q value and $Q_t^{best}(s,a)$ is the best Q value obtained in the t^{th} learning episode. In MORELA, the reward value is determined for each member of the solution vector by considering its Q value and the best Q value provided so far. The reward values come to close to the value "0" at the end of the solution process because of the structure of the reward function. In fact, a solution receives less reward when it is located closer to global optimum than the others. On the other hand, the probability of global optimum finding for further located solutions may be increased by means of providing them bigger rewards. Thus, the reward function developed may be referred to as penalty contrary to reward.

3. Numerical Experiments

An application of MORELA was carried out by solving several mathematical functions taken from the literature. However, before solving these functions, it may be essential to demonstrate the effectiveness of MORELA over RL. For this purpose, a performance comparison was conducted by solving a mathematical function. MORELA was encoded in the MATLAB for all test functions, using a computer with Intel Core i7 2.70 GHz and 8 GB of RAM. The related solution parameters for MORELA were set as follows: the environment size is taken as 20, the discounting parameter γ and the learning rate α were taken as 0.2 and 0.8, respectively. The search space parameter β was chosen according to the search domain for all test functions. The solution process was terminated when a pre-determined stopping criterion was met. The stopping criterion was properly selected for each function, and it theoretically guarantees that global optimum will be found eventually.

3.1. Comparison of MORELA and RL

The test function used to compare MORELA and RL is given in Equation (5). It has a global optimum solution of $f(x_1, x_2) = -10$ when $(x_1, x_2) = (-10, 0)$. A graph of the objective function is given in Figure 3. The convergence behaviors of MORELA and RL are illustrated in Figure 4.

$$f(x_1, x_2) = \frac{x_1}{1 + |x_2|} \tag{5}$$

As seen in Figure 4, same initial solutions were used for both algorithms to compare them realistically. Simulation results reveal that MORELA requires many fewer learning episodes than RL although both algorithms are capable of finding globally optimal solution for this function. MORELA needs only 1176 learning episodes to find optimal solution whereas RL requires 7337.

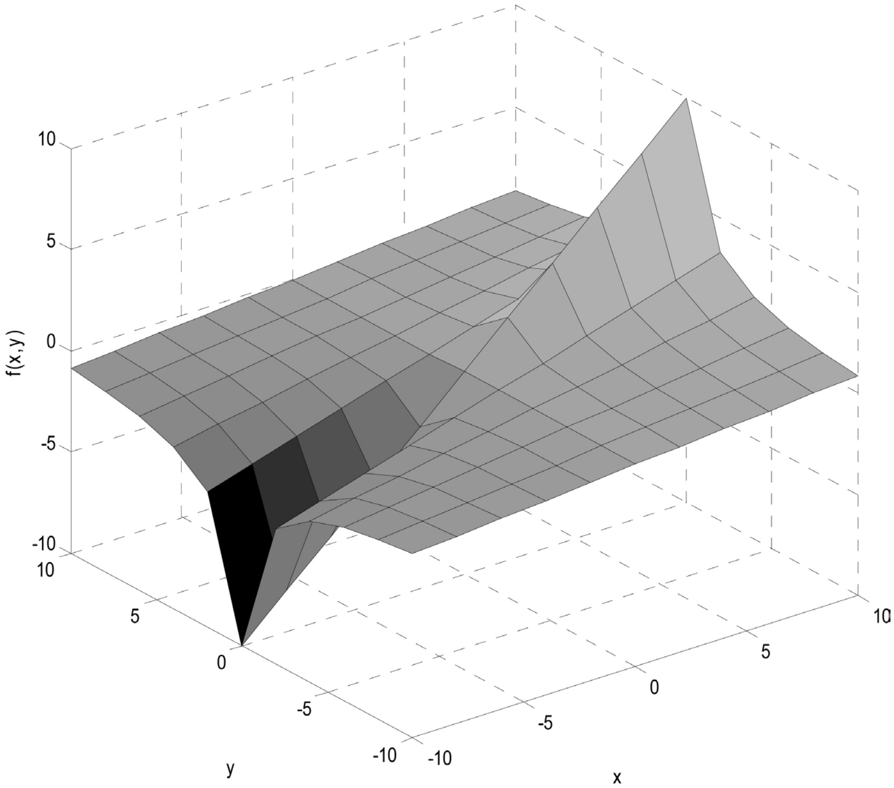


Figure 3. Objective function within the range (−10, 10).

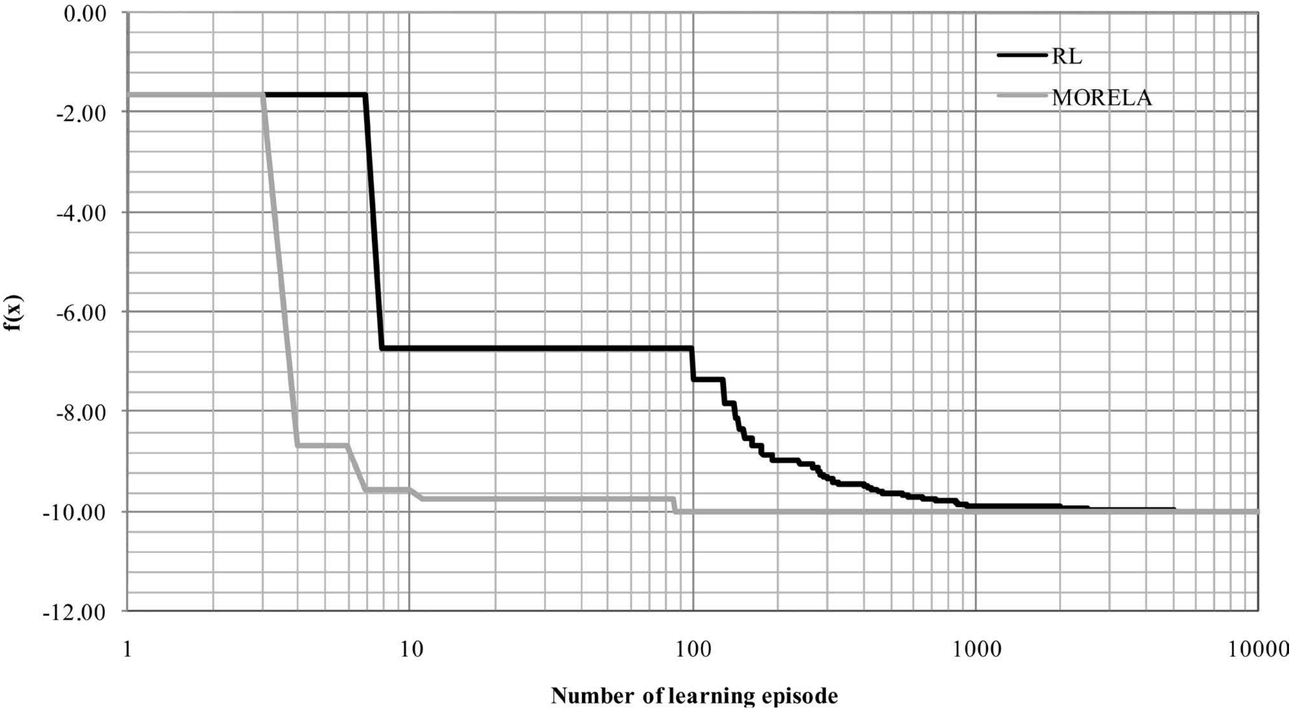


Figure 4. Performance comparisons of MORELA and RL.

3.2. Robustness Analysis

A robustness analysis for MORELA was carried out by using succeed ratio (SR) given in Equation (6).

$$SR = \frac{100 * N_s}{N_T} \quad (6)$$

where N_s is the number of successful runs which indicates that the algorithm produces the best solution at the required accuracy and N_T is the total number of runs which is set to 50 to make a fair comparison. For this experiment, a run is accepted as successful when its objective function value is around 3.5% of global optimum. The robustness analysis for MORELA, PSACO [7] and other methods [27] are given in **Table 2**. As shown, MORELA is able to find global optimum with very high success in comparison with other algorithms, except PSACO. Although PSACO produces higher success ratios than MORELA for functions F12 and F13, for given total numbers of runs, MORELA and PSACO yield same results for functions F2, F7, F9 and F16. CPSO, PSO and GA in particular yield worse results than MORELA and PSACO.

3.3. Further Comparisons of MORELA with Other Methods

To gauge the performance of MORELA against the performance of some other methods described in the literature, sixteen well-known benchmark problems were used which are given in Appendix A. Functions 6, 7, 9, 13 and 16 are taken from Shelokar *et al.* [7]. Functions 8 and 14 are adopted from Sun and Dong [28] and Chen *et al.* [29], respectively. The rest of the functions are from Baskan *et al.* [3]. **Table 3** lists algorithms compared with MORELA.

To assess the ability of MORELA, its performance was compared with 12 algorithms listed in **Table 3**. For this purpose, sixteen test functions were used based on 100 runs. The results are shown in **Table 4** in terms of the best function value, the number of learning episodes, the best solution time, the success ratio, the average number of learning episodes and the average error. The best function value is the value obtained for all runs at the required accuracy that indicates that the algorithm reached to the global optimum. The required accuracy is determined as the absolute difference between the best function value and the theoretical global optimum. For this experiment, a value of "0" was chosen as required accuracy for all test functions. The number of learning episodes and the best solution time are the number of runs and the time required to obtain the best function value, respectively. The average number of learning episodes is de-

Table 2. Results of robustness analysis.

Function	The values of SR				
	MORELA	PSACO	CPSO	PSO	GA
F2	100	100	98	100	84
F7	100	100	100	98	98
F9	100	100	100	98	98
F12	98	100	90	96	16
F13	96	98	96	26	94
F16	100	100	100	94	92

Table 3. The algorithms compared with MORELA.

Functions	Algorithm	Reference
F4-F7	SZGA	Successive zooming genetic algorithm [1]
F1-F2-F3-F4	IGARSET	Improving GA [2]
F7-F12-F13	ACO	Ant colony optimization [30]
F4-F5-F11-F12-F13-F15-F16	PSACO	Particle swarm and ant colony algorithm [7]
F5	ECTS	Enhanced continuous tabu search [31]
F10	ACORSES	Ant colony optimization [3]
F8	SA	Simulated annealing [28]
F14	RW-PSO+BOF	Random walking particle swarm optimization [29]
F5-F6-F7-F9-F11-F12-F13	GA-PSO	Genetic algorithm particle swarm optimization [8]
F4-F7-F9	GAWLS	Genetic algorithm [32]
F1-F3	HAP	Hybrid ant particle optimization algorithm [4]
F1-F4-F10	ACO-NPU	Ant colony optimization [9]
All problems	MORELA	Modified reinforcement learning algorithm (This study)

Table 4. The results of MORELA and compared algorithms.

Function	Method	Best function value	Number of learning episodes*	Best solution time (sec)	Success ratio	Average number of learning episodes*	Average error
F1	IGARSET	0	2174	0.0568	NA	2375	NA
	ACO-NPU	0	20,000	0.0590	NA	NA	NA
	HAP	2.4893e−8	100	NA	NA	NA	NA
	MORELA	0	68,760	0.8688	100	71,000	0
F2	IGARSET	−2	2400	0.0614	NA	3111	NA
	MORELA	−2	34,920	1.1790	97	36,200	0
F3	IGARSET	2.08e−27	1821	0.0666	NA	2156	NA
	HAP	2.56e−39	100	NA	NA	NA	NA
	MORELA	9.71e−40	31,620	0.9584	98	31,740	9.33e−34
F4	SZGA	2.9e−8	4000	NA	NA	NA	NA
	IGARSET	0	1004	0.0485	NA	1065	NA
	GAWLS	0	2572	NA	NA	NA	NA
	PSACO	NA	NA	NA	100	370	5.55e−17
	ACO-NPU	0	1000	0.0556	NA	NA	NA
	MORELA	0	34,340	0.5660	99	35,700	1.11e−18
F5	ECTS	NA	NA	NA	NA	338	3e−08
	PSACO	NA	NA	NA	100	190	7.69e−29
	GA-PSO	NA	NA	NA	100	206	0.00004
	MORELA	0	36,955	21.1367	100	36,980	0
F6	GA-PSO	NA	NA	NA	100	8254	0.00009
	MORELA	−1	40,680	0.6722	100	40,960	0

Continued

F7	SZGA	3	9000	NA	NA	NA	NA
	ACO	NA	NA	0.11[a]	NA	264[a]	NA
	GAWLS	3	2573	NA	NA	NA	NA
	GA-PSO	NA	NA	NA	100	25,706	0.00012
	MORELA	3	33,920	0.5344	100	37,120	0
F8	SA	−9.999994e−01	16,801	NA	NA	NA	NA
	MORELA	−1	35,640	0.5365	100	38,380	0
F9	GAWLS	−186.7309	2568	NA	NA	NA	NA
	GA-PSO	NA	NA	NA	100	96,211	0.00007
	MORELA	−186.7309[b]	16,740	0.3516	100	17,460	0
F10	ACORSES	−837.9658	1176	0.0690	NA	NA	NA
	ACO-NPU	−837.9658	750	0.0289	NA	NA	NA
	MORELA	−837.9658[b]	14,880	0.2722	100	17,500	0
F11	PSACO	NA	NA	NA	100	167	5.7061e−27
	GA-PSO	NA	NA	NA	100	95	0.00005
	MORELA	0	37,016	23.3041	100	37,856	0
F12	PSACO	NA	NA	NA	100	592	2.0755e−11
	ACO	NA	NA	0.74[a]	NA	528[a]	NA
	GA-PSO	NA	NA	NA	100	2117	0.00020
	MORELA	−3.8628[b]	13,840	0.4053	96	15,700	1.015e−13
F13	PSACO	NA	NA	NA	96	529	4.4789e−11
	GA-PSO	NA	NA	NA	100	12,568	0.00024
	ACO	NA	NA	4.10[a]	NA	1344[a]	NA
	MORELA	−3.32[c]	30,400	0.7804	96	32,200	2.3413e−16
F14	RW-PSO+BOF	NA	NA	NA	NA	NA	0[d]
	MORELA	0	33,500	0.5254	100	34,440	0
F15	PSACO	NA	NA	NA	100	1081	6.23e−22
	MORELA	0	37,010	19.8856	100	38,896	0
F16	PSACO	NA	NA	NA	100	209	2.6185e−13
	MORELA	0.3979[b]	7680	0.1413	100	7900	0

NA: Not available; [a]The average number of function evaluations of four runs and running time in units of standard time. [b]The theoretical minimum value was considered to be four digits. [c]The theoretical minimum value was considered to be two digits. [d]Mean results of more than 30 independent trials.

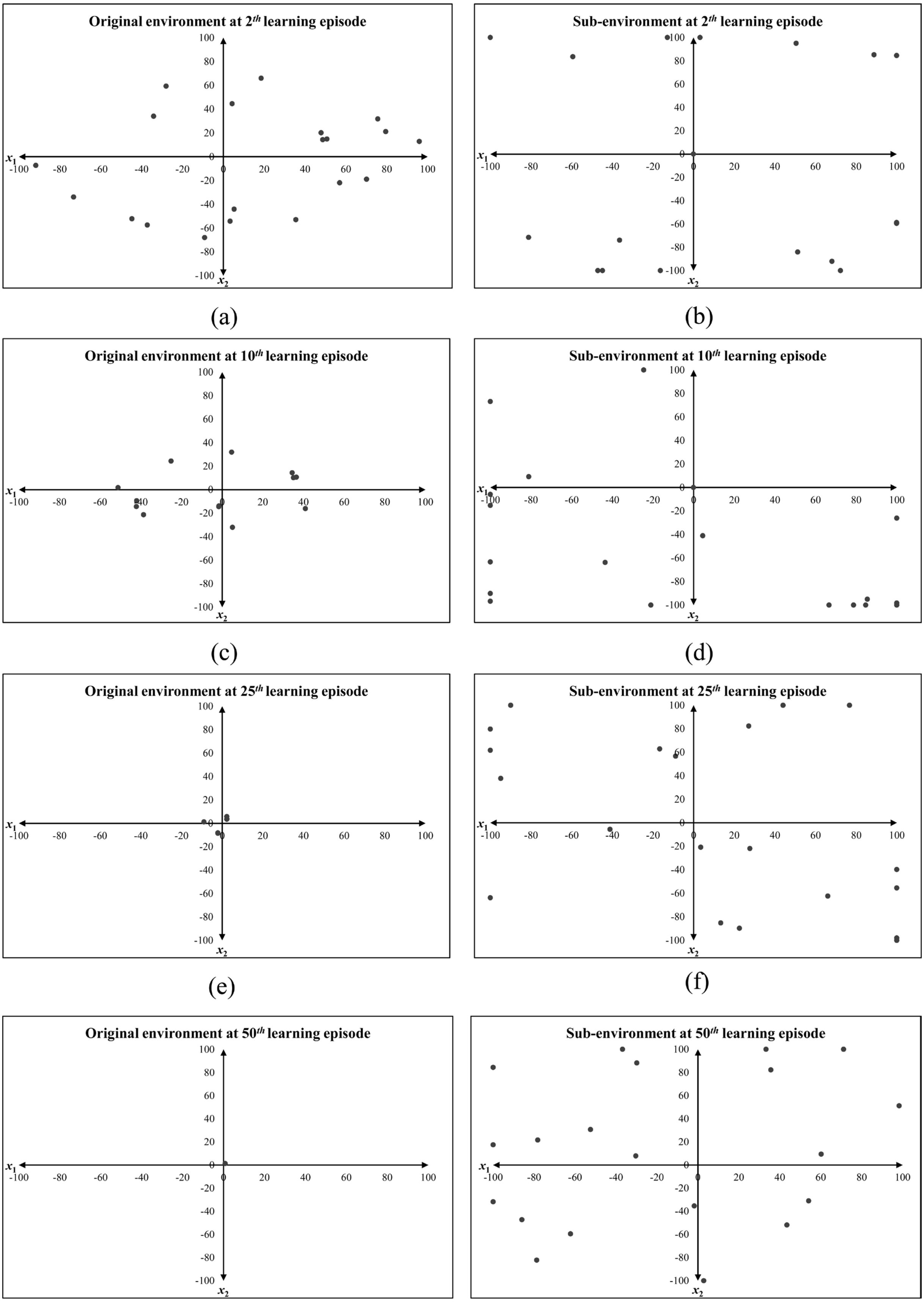
termined based on the number of successful runs in which algorithm generates the best solution for the required accuracy. Average error is defined as the average of the difference between best function value and theoretical global optimum.

The findings indicate that the MORELA showed remarkable performance for all of the test functions except F3, for which theoretical global optimum could not be found with the required accuracy, namely, 0. Although MORELA was not able to solve this function, it produces better functional value than those provided by other compared algorithms, as shown in Table 4. MORELA also produces less average error for all test functions than the other methods considered. At the same time, most of these errors are equal to 0. This means that MORELA was able to find the global optimum for each run. Therefore, MORELA may be considered to be a robust algorithm for finding global optimum for any given mathematical function. As Table 4 shows, MORELA requires a greater number of learning episodes than the other algorithms for many of the test functions, due to accuracy required of it. Although the required number of learning episodes was found to be higher than the other algorithms, it can be ignored because of the required accuracy chosen (a value of 0) for all test functions considered in this comparison. The best solution times for the functions achieved by the algorithms considered are also given in Table 4. In the meantime, investigation of the effect of environment size and corresponding algorithm parameters is the beyond the scope of this study.

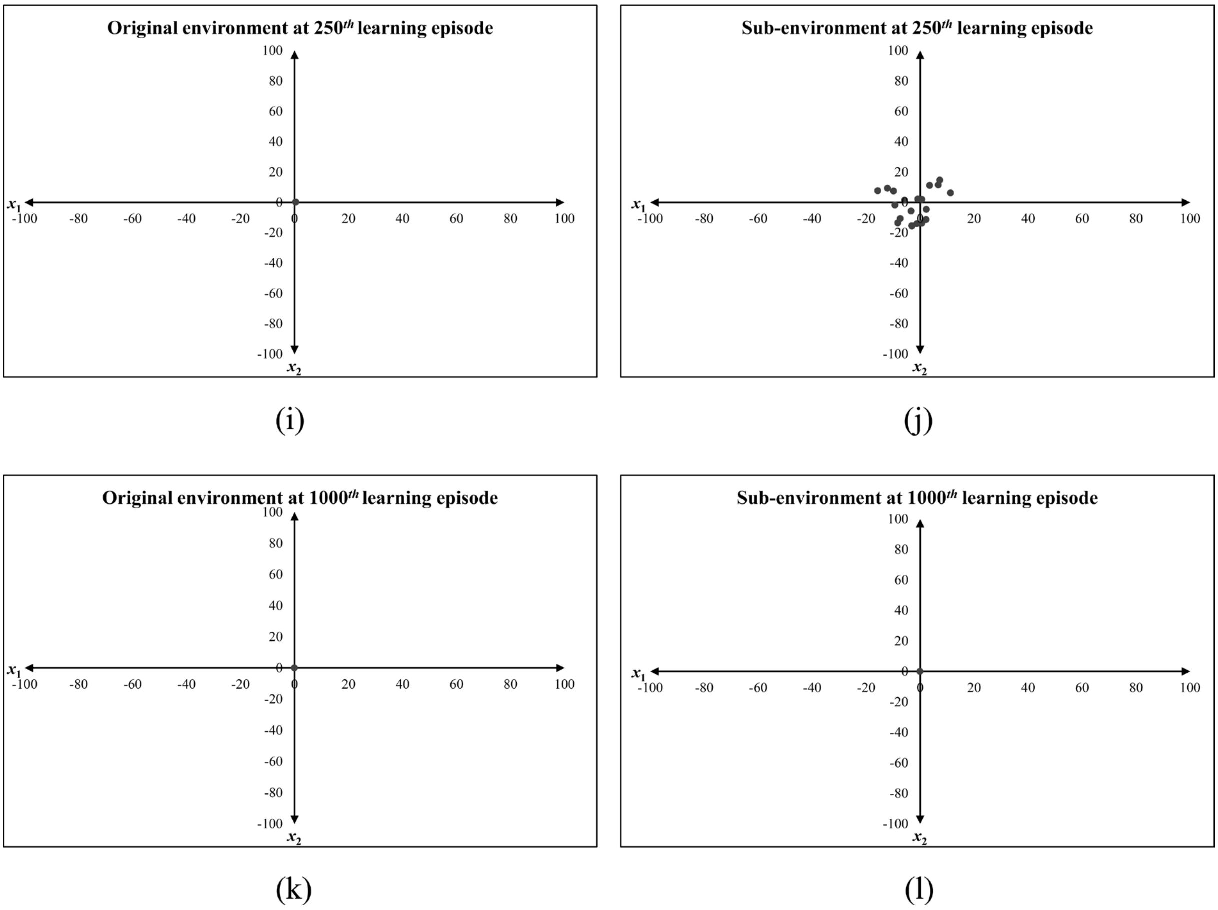
3.4. Explanation of Evolving Strategy Provided by Sub-Environment

In addition, we have used Bohachevsky function (F4) in order to better explain how the evolving population is diversified by the sub-environment in MORELA. The function of F4 has a global optimum solution of $f\left(x_1,x_2\right)=0$ when $\left(x_1,x_2\right)=0$ for the case of 2 variables and the required accuracy was chosen to be 1e−15 units for this experiment. As it can be realized from Figure 5(a) and Figure 5(b), the sub-environment is firstly generated at 2nd learning episode using Equation (3), depending on the best solution found in the previous learning episode and β value.

When the main difference between Figure 5(a) and Figure 5(b) is observed, it can be clearly seen that the solution points are stably distributed in the original environment whereas the points in the sub-environment explore global optimum near the boundaries of the solution space given as $-100\le x_1,x_2\le 100$ for this function. Although the solution points located in the original environment have a tendency to reach global optimum at the 10th learning episode as shown in Figure 5(c) and Figure 5(d), the others located in the sub-environment still continue to search global optimum near the boundaries of the solution space. This property provides to diversify the population of MORELA at each learning episode, and thus the probability of being trapped in local optimum is decreased. At 25th learning episode, the solution points in the original environment are almost close to global optimum, but the other points in the sub-environment are still dispersed in the solution space as can be seen in Figure 5(e) and Figure 5(f). Similarly, this tendency continues until MORELA reached to about 200th learning episode. After 250th learning episode, the solution points in the original environment are too close to global optimum whereas the others in the sub-environment still continues to explore new solution points around global optimum although they have a tendency to reach global optimum. Finally, at


Original environment at 2th learning episode
Sub-environment at 2th learning episode
Original environment at 10th learning episode
Sub-environment at 10th learning episode
Original environment at 25th learning episode
Sub-environment at 25th learning episode
Original environment at 50th learning episode
Sub-environment at 50th learning episode
x_1
x_2
-100
-80
-60
-40
-20
0
20
40
60
80
100

(a) (b) (c) (d) (e) (f) (g) (h)



(i) (j)

(k) (l)

Figure 5. An illustration of diversify mechanism of MORELA.

1000th learning episode, all solution points populated in the original and sub-environment reached to global optimum as seen in Figure 5(k) and Figure 5(l).

3.5. Effect of High Dimensions

The Ackley function given in Equation (7) was chosen to explore the effect of high dimensions on the search capability of MORELA. The global minimum of this function is given as $f(x)=0$ at $x=0$. The algorithm was repeated 10 times to decrease the effect of randomness. The average number of learning episodes and average objective function values were recorded with different dimensions. For this experiment, the required accuracy was chosen to be 1e−10 units.

$$AK_{(n)}(x)=-20\exp\left(-0.2*\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e \qquad (7)$$

- $-32.768\le x_i\le 32.768;\quad i=1,2,\cdots,n;\quad n=$ the number of the variables
- $x=(0,\cdots,0),\quad AK_n(x)=0$

Figure 6 represents the variation of the average objective function value according to different dimensions.

As Figure 6 shows, the MORELA shows acceptable performance even for high dimensions of the Ackley function. Average objective function values of

9.74e−11 and 9.89e−11 were obtained when the dimensions were equal to 10 and 10,000, respectively. These values are very close to each other, considering differences in dimensions. Results show that MORELA may be considered as an efficient way for finding global optimum based on required accuracy of a given mathematical function even if dimension of the problem became increased. Figure 7 illustrates the variation in the average number of learning episodes as a function of dimension for the Ackley function.

Although the average number of learning episodes increased notably with increasing dimension up to 1000, the average number of learning episodes increased very little within the dimension range from 1000 to 10,000. This experiment clearly demonstrates that the number of learning episodes required by MORELA is not apparently affected by high dimensionality.

4. Conclusions

A powerful and robust algorithm called MORELA is proposed to find global op-

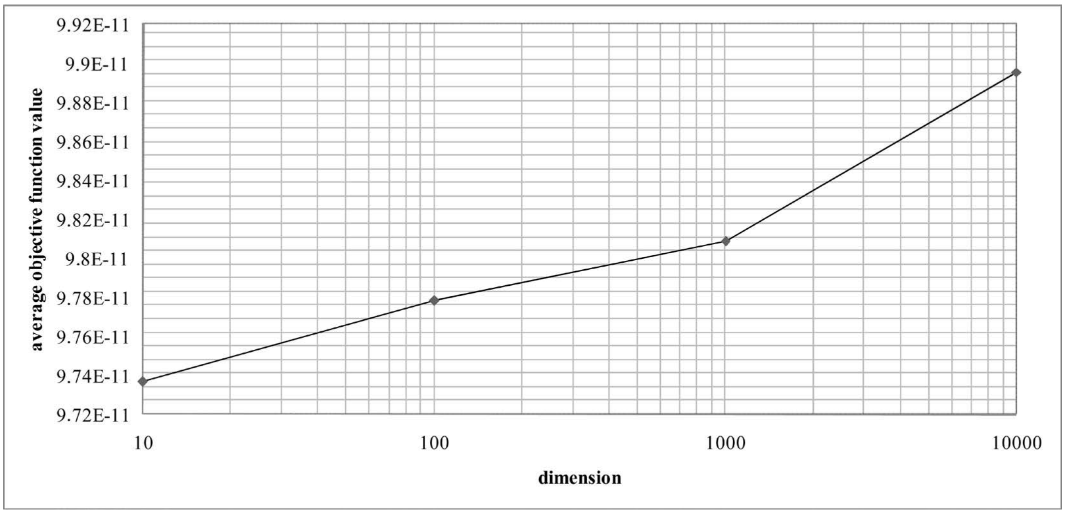


Figure 6. Average objective function values with different dimensions.

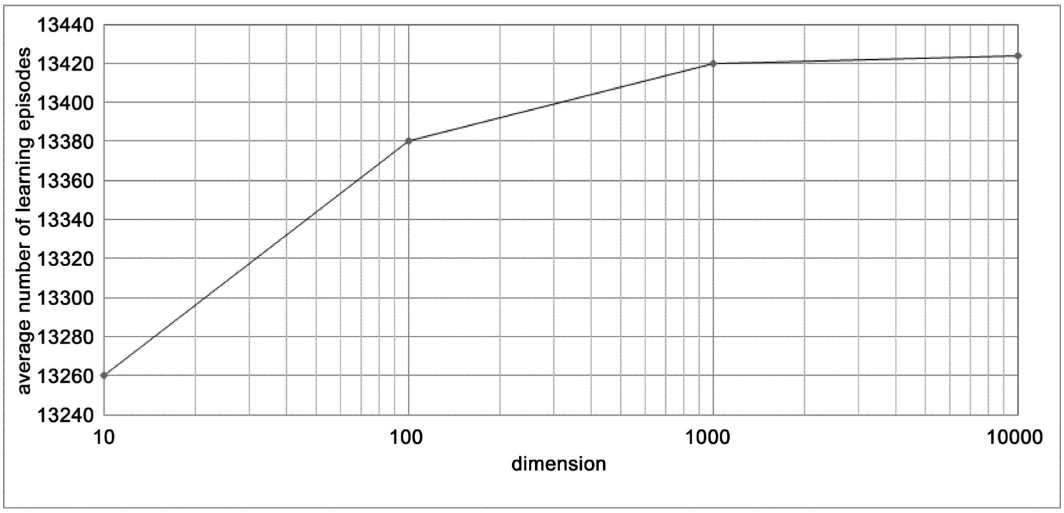


Figure 7. Average number of learning episodes with different dimensions.

timum for any given mathematical function. MORELA differs from RL based approaches by means of generating a sub-environment. Thus, this approach makes it possible to find global optimum, because it is sought both around the best solution achieved so far with the assistance of the sub-environment and the original environment.

The performance of MORELA was examined in several experiments, namely, a comparison of MORELA and RL, a robustness analysis, comparisons with other methods, explanation of evolving strategy of MORELA, and an investigation of the effect of high dimensionality. The comparison of MORELA and RL showed that MORELA requires many fewer learning episodes than RL to find global optimum for a given function. The robustness analysis revealed that MORELA is able to find global optimum with high success. MORELA was also tested on sixteen different test functions that are difficult to optimize, and its performance was compared with that of other available methods. MORELA has found global optimum for all of the test functions except F3, based on the required accuracy. Besides, the last experiment clearly shows that MORELA is not significantly affected by high dimensionality.

Finally, all numerical experiments indicate that MORELA performed well in finding global optimum of mathematical functions considered, compared to other methods. Based on the results of this study, it is expected that in future research, optimization methods based on RL will be found to possess great potential for solving various optimization problems.

References

[1] Kwon, Y.D., Kwon, S.B. and Kim, J. (2003) Convergence Enhanced Genetic Algorithm with Successive Zooming Method for Solving Continuous Optimization Problems. *Computers and Structures*, **81**, 1715-1725.

[2] Hamzacebi, C. (2008) Improving Genetic Algorithms' Performance by Local Search for Continuous Function Optimization. *Applied Mathematics and Computation*, **196**, 309-317.

[3] Baskan, O., Haldenbilen, S., Ceylan, H. and Ceylan, H. (2009) A New Solution Algorithm for Improving Performance of Ant Colony Optimization. *Applied Mathematics and Computation*, **211**, 75-84.

[4] Kıran, M.S., Gündüz, M. and Baykan, O.M. (2012) A Novel Hybrid Algorithm Based on Particle Swarm and Ant Colony Optimization for Finding the Global Minimum. *Applied Mathematics and Computation*, **219**, 1515-1521.

[5] Valian, E., Tavakoli, S. and Mohanna, S. (2014) An Intelligent Global Harmony Search Approach to Continuous Optimization Problems. *Applied Mathematics and Computation*, **232**, 670-684.

[6] Yu, S., Zhu, S., Ma, Y. and Mao, D. (2015) A Variable Step Size Firefly Algorithm for Numerical Optimization. *Applied Mathematics and Computation*, **263**, 214-220.

[7] Shelokar, P.S., Siarry, P., Jayaraman, V.K. and Kulkarni, B.D. (2007) Particle Swarm and Ant Colony Algorithms Hybridized for Improved Continuous Optimization. *Applied Mathematics and Computation*, **188**, 129-142.

[8] Kao, Y.-T. and Zahara, E. (2008) A Hybrid Genetic Algorithm and Particle Swarm Optimization for Multimodal Functions. *Applied Soft Computing*, **8**, 849-857.

[9] Seckiner, S.U., Eroglu, Y., Emrullah, M. and Dereli, T. (2013) Ant Colony Optimization for Continuous Functions by Using Novel Pheromone Updating. *Applied Mathematics and Computation*, **219**, 4163-4175.

[10] Hsieh, Y.-Z. and Su, M.-C. (2016) A *Q*-Learning-Based Swarm Optimization Algorithm for Economic Dispatch Problem. *Neural Computing and Applications*, **27**, 2333-2350. https://doi.org/10.1007/s00521-015-2070-1

[11] Samma, H., Lim, C.P. and Saleh, J.M. (2016) A New Reinforcement Learning-Based Memetic Particle Swarm Optimizer. *Applied Soft Computing*, **43**, 276-297.

[12] Walraven, E., Spaan, M.T.J. and Bakker, B. (2016) Traffic Flow Optimization: A Reinforcement Learning Approach. *Engineering Applications of Artificial Intelligence*, **52**, 203-212.

[13] Tozer, B., Mazzuchi, T. and Sarkani, S. (2017) Many-Objective Stochastic Path Finding Using Reinforcement Learning. *Expert Systems with Applications*, **72**, 371-382.

[14] Sutton, R.S. and Barto, A.G. (1998) Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA, USA; London, England.

[15] Ozan, C. (2012) Dynamic User Equilibrium Urban Network Design Based on Modified Reinforcement Learning Method. Ph.D. Thesis, The Graduate School of Natural and Applied Sciences, Pamukkale University, Denizli, Turkey. (In Turkish)

[16] Abdulhai, B. and Kattan, L. (2003) Reinforcement Learning: Introduction to Theory and Potential for Transport Applications. *Canadian Journal of Civil Engineering*, **30**, 981-991. https://doi.org/10.1139/l03-014

[17] Bazzan, A.L.C., Oliviera, D. and Silva, B.C. (2010) Learning in Groups of Traffic Signals. *Engineering Applications of Artificial Engineering*, **23**, 560-568.

[18] Vanhulsel, M., Janssens, D., Wets, G. and Vanhoof, K. (2009) Simulation of Sequential Data: An Enhanced Reinforcement Learning Approach. *Expert Systems with Applications*, **36**, 8032-8039.

[19] Kaelbling, L.P., Littman, M.L. and Moore, A.W. (1996) Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, **4**, 237-285.

[20] Liu, F. and Zeng, G. (2009) Study of Genetic Algorithm with Reinforcement Learning to Solve the TSP. *Expert Systems with Applications*, **36**, 6995-7001.

[21] Maravall, D., De Lope, J. and Martin, H.J.A. (2009) Hybridizing Evolutionary Computation and Reinforcement Learning for the Design of Almost Universal Controllers for Autonomous Robots. *Neurocomputing*, **72**, 887-894.

[22] Chen, Y., Mabu, S., Shimada, K. and Hirasawa, K. (2009) A Genetic Network Programming with Learning Approach for Enhanced Stock Trading Model. *Expert Systems with Applications*, **36**, 12537-12546.

[23] Wu, J., Xu, X., Zhang, P. and Liu, C. (2011) A Novel Multi-Agent Reinforcement Learning Approach for Job Scheduling in Grid Computing. *Future Generation Computer Systems*, **27**, 430-439.

[24] Derhami, V., Khodadadian, E., Ghasemzadeh, M. and Bidoki, A.M.Z. (2013) Applying Reinforcement Learning for Web Pages Ranking Algorithms. *Applied Soft Computing*, **13**, 1686-1692.

[25] Khamis, M.A. and Gomaa, W. (2014) Adaptive Multi-Objective Reinforcement Learning with Hybrid Exploration for Traffic Signal Control Based on Cooperative Multi-Agent Framework. *Engineering Applications of Artificial Intelligence*, **29**, 134-151.

[26] Ozan, C., Baskan, O., Haldenbilen, S. and Ceylan, H. (2015) A Modified Reinforce-

ment Learning Algorithm for Solving Coordinated Signalized Networks. *Transportation Research Part C*, **54**, 40-55.

[27] Liu, B., Wang, L., Jin, Y.-H., Tang, F. and Huang, D.-X. (2005) Improved Particle Swarm Optimization Combined with Chaos. *Chaos*, *Solitons and Fractals*, **25**, 1261-1271.

[28] Sun, W. and Dong, Y. (2011) Study of Multiscale Global Optimization Based on Parameter Space Partition. *Journal of Global Optimization*, **49**, 149-172. https://doi.org/10.1007/s10898-010-9540-x

[29] Chen, C., Chang, K. and Ho, S. (2011) Improved Framework for Particle Swarm Optimization: Swarm Intelligence with Diversity-Guided Random Walking. *Expert Systems and Applications*, **38**, 12214-12220.

[30] Toksarı, M.D. (2009) Minimizing the Multimodal Functions with Ant Colony Optimization Approach. *Expert Systems and Applications*, **36**, 6030-6035.

[31] Chelouah, R. and Siarry, P. (2000) Tabu Search Applied to Global Optimization. *European Journal of Operational Research*, **123**, 256-270.

[32] Tutkun, N. (2009) Optimization of Multimodal Continuous Functions Using a New Crossover for the Real-Coded Genetic Algorithms. *Expert Systems and Applications*, **36**, 8172-8177.

Appendix A

F1:

Rosenbrock (2 *variables*)

$$f(x_1,x_2)=\left(100*\left(x_1-x_2^2\right)^2\right)+\left(1-x_1\right)^2$$

- global minimum: $(x_1,x_2)=1,\ f(x_1,x_2)=0$

F2:

(2 *variables*)

$$f(x_1,x_2)=x_1^2+x_2^2-\cos(18x_1)-\cos(18x_2)$$

- $-1\le x_1,x_2\le 1$
- global minimum: $(x_1,x_2)=(0,0),\ f(x_1,x_2)=-2$

F3:

(2 *variables*)

$$f(x_1,x_2)=\frac{(x_1-3)^8}{1+(x_1-3)^8}+\frac{(x_2-3)^4}{1+(x_2-3)^4}$$

- global minimum: $(x_1,x_2)=3,\ f(x_1,x_2)=0$

F4:

Bohachevsky (2 *variables*)

$$f(x_1,x_2)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$$

- $-100\le x_1,x_2\le 100$
- global minimum: $(x_1,x_2)=0,\ f(x_1,x_2)=0$

F5:

De Jong (3 *variables*)

$$f(x)=\sum_{i=1}^{n}x_i^2$$

- $-5.12\le x_i\le 5.12; i=1,2,\cdots,n$
- global minimum: $x=(0,0,\cdots,0),\ f(x)=0$

F6:

Easom (2 *variables*)

$$f(x_1,x_2)=-\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$$

- $-100\le x_1,x_2\le 100$
- global minimum: $(x_1,x_2)=(\pi,\pi),\ f(x_1,x_2)=-1$

F7:

Goldstein-Price (2 *variables*)

$$f(x,y)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$$
$$*\left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$$

- $-2\le x_1,x_2\le 2$
- global minimum: $(x_1,x_2)=(0,-1),\ f(x_1,x_2)=3$

F8:

Drop wave (2 *variables*)

$$f\left(x_1, x_2\right) = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{\frac{1}{2}\left(x_1^2+x_2^2\right)+2}$$

- $-5.12 \le x_1, x_2 \le 5.12$
- global minimum: $\left(x_1, x_2\right)=\left(0,0\right), \left(x_1, x_2\right)=-1$

F9:

Shubert (2 *variables*)

$$f\left(x_1, x_2\right)=\sum_{i=1}^{5} i\cdot\cos\left(\left(i+1\right)x_1+1\right)*\sum_{i=1}^{5} i\cdot\cos\left(\left(i+1\right)x_2+1\right)$$

- $-10 \le x_1, x_2 \le 10$
- 18 global minima $f\left(x_1, x_2\right)=-186.7309$

F10:

Schwefel (2 *variables*)

$$f\left(x\right)=\sum_{i=1}^{n} -x_i * \sin\sqrt{\left|x_i\right|}$$

- $-500 \le x_1, x_2 \le 500$
- global minimum: $\left(x_1, x_2\right)=\left(420.9687, 420.9687\right), f\left(x_1, x_2\right)=-n*418.9829$

F11:

Zakharov (2 *variables*)

$$f\left(x\right)=\sum_{i=1}^{n} x_i^2+\left(\sum_{i=1}^{n} 0.5ix_i\right)^2+\left(\sum_{i=1}^{n} 0.5ix_i\right)^4$$

- $-5 \le x_i \le 10, i=1,2,\cdots,n$
- global minimum: $x=\left(0,0,\cdots,0\right), f\left(x\right)=0$

F12:

Hartman (3 *variables*)

$$f\left(x\right)=-\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} \alpha_{ij}\left(x_j-p_{ij}\right)^2\right]$$

- $0 \le x_j \le 1, j=1,\cdots,3$
- global minimum: $x=\left(0.11, 0.555, 0.855\right), f\left(x\right)=-3.8628$

i	α_{i1}	α_{i2}	α_{i3}	c_i	p_{i1}	p_{i2}	p_{i3}
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8742	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

F13:

Hartman (6 *variables*)

$$f\left(x\right)=-\sum_{i=1}^{4}c_i \exp\left[-\sum_{j=1}^{n}\alpha_{ij}\left(x_j-p_{ij}\right)^2\right]$$

- $0\le x_j\le 1, j=1,2,\cdots,6$
- global minimum: $x=\left(0.201,0.150,0.477,0.275,0.311,0.657\right), f\left(x\right)=-3.32$

i	α_{i1}	α_{i2}	α_{i3}	α_{i4}	α_{i5}	α_{i6}	c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

i	p_{i1}	p_{i2}	p_{i3}	p_{i4}	p_{i5}	p_{i6}
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

F14:

Rastrigin (2 *variables*)

$$f\left(x\right)=10n+\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)\right]$$

- $-5.12\le x_i\le 5.12, i=1,2,\cdots,n$
- global minimum: $\left(x_1,x_2\right)=\left(0,0\right), f\left(x\right)=0$

F15:

Griewank (8 *variables*)

$$f\left(x\right)=\sum_{i=1}^{n}x_i^2/4000-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

- $-300\le x_i\le 600, i=1,2,\cdots,n$
- global minimum: $x=\left(0,\cdots,0\right), f\left(x\right)=0$

F16:

Branin (2 *variables*)

$$f\left(x\right)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos\left(x_1\right)+10$$

- $-5\le x_1\le 10, 0\le x_2\le 15$
- three global minimum: $\left(-\pi,12.275\right),\left(\pi,2.275\right),\left(3\pi,2.475\right), f\left(x\right)=\frac{5}{4\pi}$