Article

Not peer-reviewed version

Evolutionary Reinforcement Learning of Neural Network Controller for Acrobot Task — Part4: Particle Swarm Optimization

Hidehiko Okada *

Posted Date: 4 March 2024

doi: 10.20944/preprints202403.0109.v1

Keywords: swarm intelligence; particle swarm optimization; neural network; neuroevolution; reinforcement learning

Article

Evolutionary Reinforcement Learning of Neural Network Controller for Acrobot Task—Part4: Particle Swarm Optimization

Hidehiko Okada

Faculty of Information Science and Engineering, Kyoto Sangyo University, Japan; hidehiko@cc.kyoto-su.ac.jp

Abstract: Evolutionary algorithms and swarm intelligence algorithms find applicability in reinforcement learning of neural networks due to their independence from gradient-based methods. To achieve successful training of neural networks using these algorithms, careful considerations must be made to select appropriate algorithms due to the availability of various algorithmic variations. In Part1, 2 and 3, the author previously reported experimental evaluations on Evolution Strategy, Genetic Algorithm, and Differential Evolution for reinforcement learning of neural networks, utilizing the Acrobot control task. This article constitutes Part4 of the series of comparative research. In this study, Particle Swarm Optimization is adopted as an instance of major swarm intelligence algorithms. The experimental result shows that PSO performed worse than all of DE, GA and ES. The difference between PSO and DE was statistically significant ($p<.01$). In addition, PSO exhibited lower capability in exploring solutions in high-dimensional search spaces than DE, GA, and ES did. A larger swarm size compensated for the weakness of PSO in global exploration, thus making itself more beneficial than a larger number of swarm search iterations.

Keywords: swarm intelligence; particle swarm optimization; neural network; neuroevolution; reinforcement learning

1. INTRODUCTION

Neural networks can be effectively trained using gradient-based methods for supervised learning tasks, where labeled training data are available. However, when it comes to reinforcement learning tasks where labeled training data are not provided, neural networks demand the utilization of gradient-free training algorithms. Evolutionary algorithms [1-5] and swarm intelligence algorithms [6-10] find applicability in the reinforcement learning of neural networks due to their independence from gradient-based methods.

Particle Swarm Optimization [11-13], Ant Colony Optimization [14,15], Artificial Bee Colony [16,17], and Cuckoo Search [18,19] are representative swarm intelligence algorithms. To achieve successful training of neural networks using swarm algorithms, careful considerations must be made regarding: i) selecting appropriate algorithms due to the availability of various algorithmic variations, and ii) designing hyperparameters as they significantly impact performance. The author previously reported experimental evaluations on major evolutionary algorithms, Evolution Strategy [20,21], Genetic Algorithm [22-25], and Differential Evolution [26-28], for reinforcement learning of neural networks, utilizing the Acrobot task [29-31]. This study applies PSO to the same learning task and compares its performance with those of ES, GA and DE.

2. ACROBOT CONTROL TASK

As a task that requires reinforcement learning to solve, this study employs Acrobot control task provided at OpenAI Gym. Figure 1 shows a screenshot of the system. This task is the same as that in the previous studies; details on this task were described in Part1 [29]. The same method for scoring fitness of a neural network controller (eq.(1) in Part1) is employed again in this study.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 March 2024 doi:10.20944/preprints202403.0109.v1

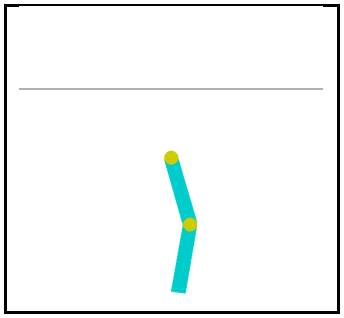


Figure 1. Acrobot system[1].

3. NEURAL NETWORKS

In the previous studies using DE, GA, and ES [29-31], the author employed a three-layered feedforward neural network known as a multilayer perceptron (MLP [32,33]) as the controller. The same MLP is utilized again in this study. Figure 2 illustrates the topology of the MLP. The feedforward calculations were described in Part1 [29]. In all of this study and the previous ones, the MLP serves as the policy function: action(t) = F(observation(t)). The input layer consists of six units, each corresponding to the values obtained by an observation. To ensure the input value falls within the range [-1.0, 1.0], the angular velocity of θ_1 (θ_2) is divided by 4π (9π). The output layer comprises one unit, and its output value is applied as the torque to the joint.

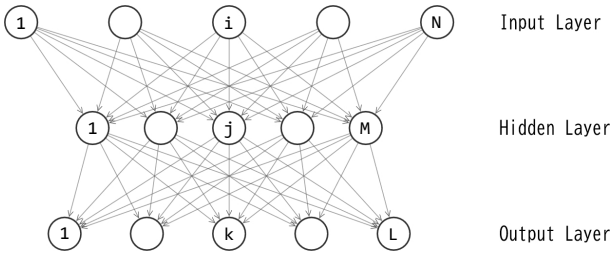


Figure 2. Topology of the MLP.

4. TRAINING OF NEURAL NETWORKS BY PARTICLE SWARM OPTIMIZATION

The three-layered perceptron depicted in Figure 2 includes M+L unit biases and NM+ML connection weights, resulting in a total of M+L+NM+ML parameters. Let D represent the quantity M+L+NM+ML. For this study, the author sets N=6 and L=1, leading to D=8M+1. The training of this perceptron is essentially an optimization of the D-dimensional real vector. Let $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ denote the D-dimensional vector, where each x_i corresponds to one of the D parameters in the perceptron. By applying the value of each element in $\mathbf{x}$ to its corresponding connection weight or unit bias, the feedforward calculation (described by eqs. (2)-(6) in Part1 [29]) can be processed.

In this study, PSO is applied to optimize the D-dimensional vector $\mathbf{x}$. PSO represents one of the swarm intelligence algorithms, which are characterized by being population-based stochastic search algorithms. PSO utilize $\mathbf{x}$ as a particle position in the D-dimensional search space. The fitness of $\mathbf{x}$ is evaluated based on eq. (1) described in Part1 [29]. Figure 3 illustrates the PSO process. Steps 1-3 are the same as those in Evolution Strategy, which are described in Part1 [29]. Step 1 initializes vectors $\mathbf{x}^1$, $\mathbf{x}^2$, …, $\mathbf{x}^\lambda$ randomly within a predefined range, denoted as $[min, max]^D$, where λ represents the number of particles (the swarm size). In Step 2, the values in each vector $\mathbf{x}^s$ (s=1, 2, ..., λ) are fed into the MLP, which subsequently controls the Acrobot system for a single episode consisting of 200 time steps. The fitness of $\mathbf{x}^s$ is evaluated based on the outcome of the episode. In Step 3, the iterative training loop concludes upon meeting a preset condition. A straightforward example of such a condition is reaching the limit number of fitness evaluations. In Step 4, the personal best (Pbest) of

[1] https://www.gymlibrary.dev/environments/classic_control/acrobot/

 doi:10.20944/preprints202403.0109.v1

each particle and the global best (Gbest) in the swarm are updated according to their fitness scores. The Pbest of a particle represents the position vector that has achieved the highest fitness score up to the current iteration for that specific particle. On the other hand, the Gbest represents the position vector with the highest fitness score among all the Pbests in the swarm. Let us denote each Pbest as $\mathbf{p}^s$ and the Gbest as $\mathbf{g}$, respectively. In Step 5, the velocity of each particle is updated. Let $\mathbf{v}^s = (v_1^s, v_2^s, \ldots, v_D^s)$ represents the velocity for the s-th particle (s=1, 2, ..., λ). The velocity $\mathbf{v}^s$ is updated by eq. (1), where w denotes the inertia weight, c_p is the Pbest coefficient, and c_g is the Gbest coefficient. Additionally, r_p and r_g are uniformly distributed random numbers within the interval [0,1]. In Step 6, each particle moves in the search space according to its velocity. The position vector $\mathbf{x}^s$ is updated by eq. (2).

$$\mathbf{v}^s \leftarrow w\mathbf{v}^s + c_p r_p(\mathbf{p}^s - \mathbf{x}^s) + c_g r_g(\mathbf{g} - \mathbf{x}^s) \quad (1)$$

$$\mathbf{x}^s \leftarrow \mathbf{x}^s + \mathbf{v}^s \quad (2)$$

Step 1. Initialization
Step 2. Fitness Evaluation
Step 3. Conditional Termination
Step 4. Updates of Pbests and Gbest
Step 5. Updates of Particle Velocities
Step 6. Updates of Particle Positions
Step 7. Goto Step 2

Figure 3. Process of Particle Swarm Optimization.

5. EXPERIMENT

In the previous studies using DE, GA, and ES, the number of fitness evaluations included in one experiment trial was set to 5,000 [29-31]. The number of new offsprings generated per generation, λ, was either of (a) 10 and (b) 50. The number of generations was 500 for (a) and 100 for (b) respectively. The total number of fitness evaluations were 10 × 500 = 5,000 for (a) and 50 × 100 = 5,000 for (b). The experiment using PSO in this study employed the same settings to fairly compare PSO with DE, GA and ES. The hyperparameter configurations for PSO are shown in Table 1. The values of w, c_p and c_g adopted in this study are those suggested by Eberhart and Shi [34]. The domain of particle position vectors, $[min, max]^D$, was kept consistent with the previous experiments [29-31], i.e., [-10.0, 10.0]^D. The number of hidden units, M, was also consistently set to the four variations: 4, 8, 16, and 32. An MLP with either of 4, 8, 16, or 32 hidden units underwent independent training 11 times. Table 2 presents the best, worst, average, and median fitness scores of the trained MLPs across the 11 trials. Each of the two hyperparameter configurations (a) and (b) in Table 1 was applied.

Table 1. PSO Hyperparameters.

Hyperparameters	(a)	(b)
Number of particles (λ)	10	50
Iterations	500	100
Fitness evaluations	10×500=5,000	50×100=5,000
Inertia weights (w)	0.729	0.729
Pbest coefficient (c_p)	1.49445	1.49445
Gbest coefficient (c_g)	1.49445	1.49445

Table 2. Fitness Scores among 11 Runs.

	M	Best	Worst	Average	Median
(a)	**4**	0.469	0.336	0.408	0.417
	8	0.450	0.269	0.401	0.435

 doi:10.20944/preprints202403.0109.v1

	16	0.467	0.333	0.424	0.439
	32	0.465	0.314	0.418	0.425
(b)	**4**	0.460	0.416	0.440	0.436
	8	0.463	0.396	0.437	0.436
	16	0.458	0.351	0.429	0.439
	32	0.453	0.391	0.425	0.424

Comparing the scores in Table 2 between configurations (a) and (b), 12 of the 16 scores (12/16 = 75%) obtained using configuration (b) are larger than their corresponding scores obtained using configuration (a). The Wilcoxon signed rank test confirmed that the difference between scores of configuration (b) and those of (a) was statistically significant ($p < .01$). This result is consistent with the previous study [29] in which, instead of PSO, ES was adopted as the optimization algorithm. Configuration (a) promotes exploitation in the later stage of search due to the large number of iterations, and configuration (b) promotes exploration in the early stage of search due to the large number of particles. PSO is criticized for its tendency towards premature convergence due to the influence of Gbest, excelling in local exploitation but struggling with global exploration. This weakness, shared with ES, lies in its limited capability for global exploration. Configuration (b) can compensate for this weakness, thus making configuration (b) more beneficial than configuration (a) for both PSO and ES.

Next, upon comparing the fitness scores obtained using configuration (b) among the four variations of M (the number of hidden units), it is observed that the four scores with M=32 are found to be smaller than those with M=4, 8, and 16. No clear superiority is observed among the scores with M=4, 8, and 16. The Wilcoxon rank sum test confirmed that the scores with M=32 are significantly smaller than those with M=4 ($p < .05$). While no significant differences were observed between the scores with M=32 and those with M=8, 16 ($p= .12$ and .28 respectively), these p-values indicate that the scores with M=32 were inferior to those with M=8, 16. The scores with M=4 are comparable to those with M=8 ($p= .47$) and superior to those with M=16 ($p= .28$). From the perspective of model size and performance, M=4 can be considered the most desirable in this study. Previous studies [29-31] employing DE, GA, and ES indicated that M=8, rather than M=4, was the most desirable, with statistically significant differences observed ($p < .05$ for DE and GA, $p < .01$ for ES). Based on the results of the past studies and the current study, it is suggested that PSO exhibits lower capability in exploring solutions in high-dimensional search spaces than DE, GA, and ES did. Consequently, M=4, where the dimensionality D is minimum, is deemed preferable. The performance differences between PSO and DE, GA, ES will be further discussed in the subsequent Section 6.

Figure 4 presents learning curves of the best, median, and worst runs among the 11 trials where the configuration is (b). Note that the horizontal axis of these graphs is in a logarithmic scale. The shapes of these graphs are similar to those obtained using DE, GA, and ES [29-31]. However, a significant difference is that in the case of PSO, the final fitness scores after the 5000 evaluations are scattered between trials. For all M=4, 8, 16, 32, the differences between the best and the worst trials are large. In previous studies using DE, GA, and ES, these differences were much smaller than those in the case of PSO. In other words, when using DE, GA, and ES, good solutions were stably found without depending on the differences in random initial solutions. However, the results of this experiment indicate that PSO is prone to variability in solution exploration performance depending on the differences in random initial solutions.

 doi:10.20944/preprints202403.0109.v1

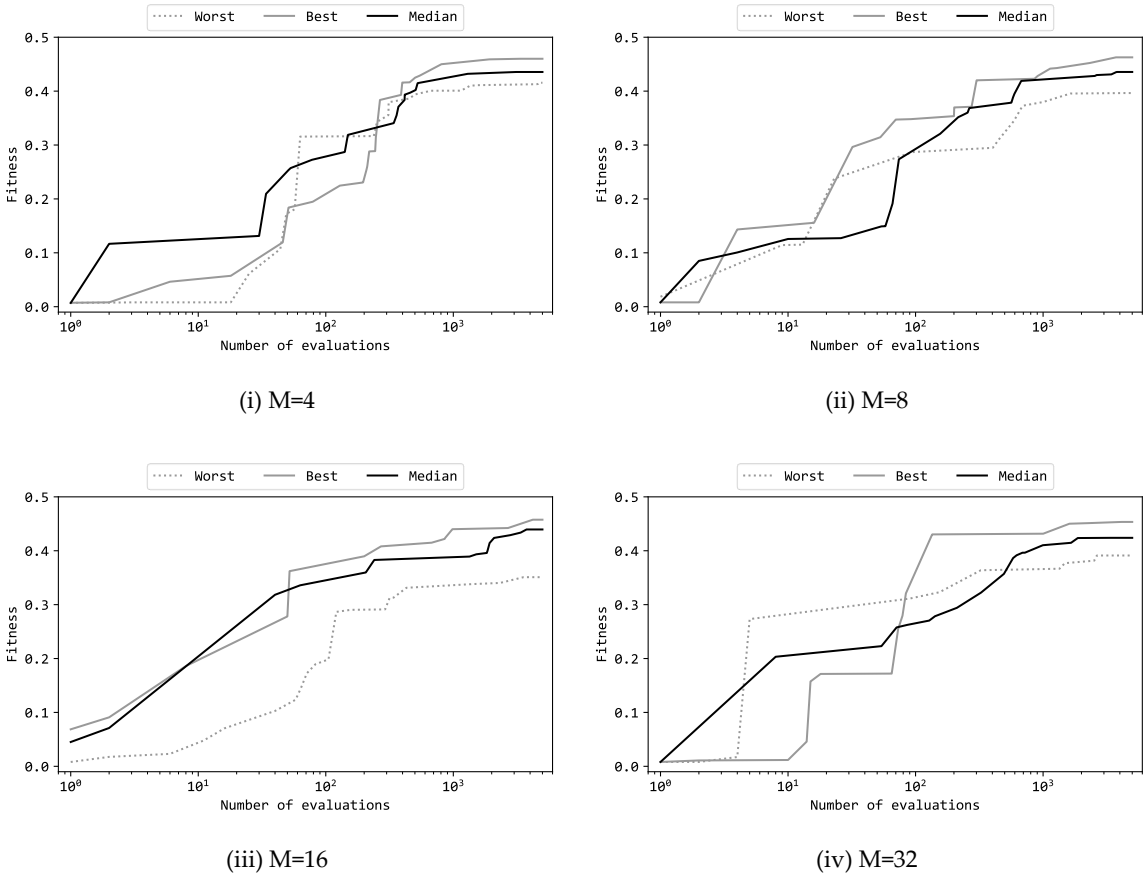
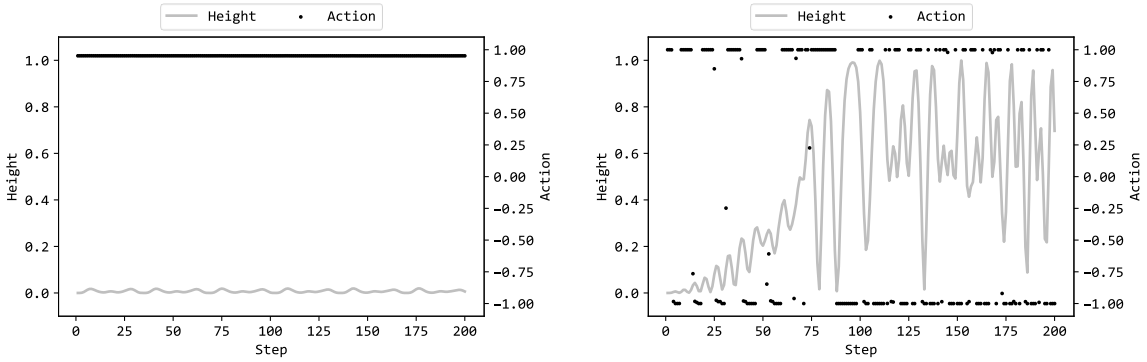


Figure 4. Learning curves of MLP with M hidden units.

Figure 5(a) illustrates the actions by the MLP and the heights $p_y(t)$ (see eq. (1) in Part1 [29]) in the 200 steps prior to training, while Figure 5(b) displays the corresponding actions and heights after training. Supplementary videos are provided which demonstrate the motions of the chain[2,3]. In this scenario, the MLP employed 4 hidden units, and the configuration (b) in Table 1 was utilized. In the previous articles [29-31] which reported the corresponding results by DE, GA and ES, the actions and the heights are illustrated in Figures 5(a)(b). Comparing the four Figures 5(a) for PSO, DE, GA, and ES, in each case, the MLP before the training continuously output values close to 1.0 (or -1.0) for the entire 200 steps, resulting in the Acrobot chain being unable to move much, and the heights $p_y(t)$ remaining almost unchanged from its initial value of 0.0. On the other hand, comparing the four Figures 5(b) for PSO, DE, GA, and ES, the shapes of the graphs are similar. The graph for PSO is particularly similar to the graphs for GA and DE: after the heights $p_y(t)$ exceeded 0.5 around 75 steps, the number of times $p_y(t)$ fell below 0.5 was small. This indicates that the trained MLP was able to skillfully control the torque to avoid the free end of the Acrobot chain from falling back to a low position after lifting it to a high position. However, even in the case of PSO, it was not possible for the trained MLP to maintain the entire chain in an inverted state and bring the free end of the chain to the highest position at 1.0, which was the same as the results with DE, GA, and ES.

[2] http://youtu.be/SsemOB3lZ6I
[3] http://youtu.be/jypHSup8h2c

 doi:10.20944/preprints202403.0109.v1



(a) Before the training. (b) After the training.

Figure 5. MLP actions and the height $p_y(t)$ in an episode.

6. STATISTICAL TEST TO COMPARE PSO WITH DE, GA, AND ES

The author conducted a Wilcoxon signed rank test to investigate whether there was a statistically significant difference between the fitness scores obtained from the previous experiments using DE, GA or ES and those from the experiment using PSO in this study. The data used for this test included the 32 values reported in Table 2 of Part1 [29] (the data obtained using ES), those in Table 2 of Part2 [30] (the data obtained using GA), those in Table 2 of Part3 [31] (the data obtained using DE), and those in Table 2 of this article (the data obtained using PSO). The test result indicates that the fitness scores by PSO are significantly smaller than those by DE ($p < .01$). Although the differences between the fitness scores by PSO and those by GA or ES are not statistically significant, the p-values reveals that the scores by PSO are smaller than those by GA ($p=.25$) and those by ES ($p=.07$). These test results indicate that, on the MLP training task in this study, PSO performs significantly worse than DE and also performs worse than GA and ES. It is known that particles controlled by PSO are prone to being trapped in local optima, and various improvements have been proposed to address this drawback (e.g. [35]). Future research will be necessary to evaluate the effectiveness of these improvements on this training task.

7. Conclusion

In this study, Particle Swarm Optimization was applied to the reinforcement learning of a neural network controller for the Acrobot task, and the result was compared with that previously reported in Part1/2/3 [29-31] where ES/GA/DE was applied to the same task respectively. The findings from this study are summarized as follows:

(1) The statistical tests revealed that PSO performed worse than all of DE, GA and ES. The difference between PSO and DE was statistically significant ($p<.01$).
(2) The experiment in this study employed two configurations, which are consistent with the previous studies reported in Part1-3: maintaining a fixed number of 5000 fitness evaluations, (a) a greater number of particles, suitable for early-stage global exploration, and (b) a greater number of iterations, suitable for late-stage local exploitation. A comparative analysis of the results revealed that configuration (b) contributed significantly better than configuration (a). Configuration (b) can compensate for the limited capability of PSO in global exploration, thus making itself more beneficial than configuration (a). This result is consistent with the previous study in which ES was adopted [29].
(3) Four different numbers of units in the hidden layer of the multilayer perceptron were compared: 4, 8, 16, and 32. The experimental results revealed that 4 units were found to be the optimal choice from the perspective of the trade-off between performance and model size. This result does not align with previous studies: 8 units were the best in the cases of DE, GA, and ES. PSO exhibited lower capability in exploring solutions in high-dimensional search spaces than DE, GA, and ES did.

 doi:10.20944/preprints202403.0109.v1

The author plans to further apply and evaluate another evolutionary/swarm algorithms to the same task and compare the performance with those by PSO, DE, GA, and ES.

Acknowledgments: The author conducted this study under the Official Researcher Program of Kyoto Sangyo University.

References

1. Bäck, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. Evolutionary computation, 1(1), 1-23.
2. Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. IEEE transactions on neural networks, 5(1), 3-14.
3. Bäck, T. (1996). Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.
4. Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. IEEE transactions on evolutionary computation, 3(2), 124-141.
5. Eiben, Á. E., & Smith, J. E. (2015). Introduction to evolutionary computing. Springer-Verlag Berlin Heidelberg.
6. Bari, A., Zhao, R., Pothineni, J.S., Saravanan, D. (2023). Swarm intelligence algorithms and applications: an experimental survey. Advances in swarm intelligence, ICSI 2023, Lecture notes in computer science, 13968, Springer.
7. Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: applications and trends. IEEE/CAA journal of automatica sinica, 8(10), 1627-1643.
8. Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: algorithms and applications. Swarm and evolutionary computation, 33, 1-17.
9. Payal, M., Kumar, A., & Díaz, V. G. (2020). A fundamental overview of different algorithms and performance optimization for swarm intelligence. Swarm intelligence optimization: algorithms and applications, 1-19.
10. Chu, SC., Huang, HC., Roddick, J.F., Pan, JS. (2011). Overview of algorithms for swarm intelligence. Computational collective intelligence, technologies and applications. ICCCI 2011. Lecture notes in computer science, 6922, Springer.
11. Kennedy J. & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International conference on neural networks, 4, 1942-1948.
12. Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. MHS'95. Proceedings of the sixth international symposium on micro machine and human science, 39-43.
13. Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. IEEE international conference on evolutionary computation proceedings, 69-73.
14. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE transactions on systems, man, and cybernetics, part b (cybernetics), 26(1), 29-41.
15. Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), 53-66.
16. Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. International fuzzy systems association world congress, 789-798.
17. Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39, 459-471.
18. Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. IEEE world congress on nature & biologically inspired computing, 210-214.
19. Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. International journal of mathematical modelling and numerical optimisation, 1(4), 330-343.
20. Schwefel, H. P. (1984). Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. Annals of operations research, 1(2), 165-167.
21. Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies – a comprehensive introduction. Natural computing, 1, 3-52.
22. Goldberg, D.E., Holland, J.H. (1988). Genetic algorithms and machine learning. Machine learning, 3, 95-99, doi: 10.1023/A:1022602019183.
23. Holland, J. H. (1992). Genetic algorithms. Scientific American, 267(1), 66-73.
24. Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.
25. Sastry, K., Goldberg, D., & Kendall, G. (2005). Genetic algorithms. Search methodologies: introductory tutorials in optimization and decision support techniques, 97-125.
26. Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11, 341-359.

27. Price, K., Storn, R. M., & Lampinen, J. A. (2006). Differential evolution: a practical approach to global optimization. Springer science & business media.
28. Das, S., & Suganthan, P. N. (2010). Differential evolution: a survey of the state-of-the-art. IEEE transactions on evolutionary computation, 15(1), 4-31.
29. Okada, H. (2023). Evolutionary reinforcement learning of neural network controller for Acrobot task – Part1: evolution strategy. Preprints.org, doi: 10.20944/preprints202308.0081.v1.
30. Okada, H. (2023). Evolutionary reinforcement learning of neural network controller for Acrobot task – Part2: genetic algorithm. Preprints.org, doi: 10.20944/preprints202310.0852.v1.
31. Okada, H. (2024). Evolutionary reinforcement learning of neural network controller for Acrobot task – Part3: differential evolution. Preprints.org, doi: 10.20944/preprints202402.0145.v1.
32. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. Parallel distributed processing: explorations in the microstructure of cognition, vol.1: foundations. MIT Press, 318-362.
33. Collobert, R., & Bengio, S. (2004). Links between perceptrons, MLPs and SVMs. Proceedings of the twenty-first international conference on machine learning (ICML 04), doi: 10.1145/ 1015330.1015415.
34. Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the 2000 congress on evolutionary computation (CEC00), 1, 84-88.
35. Bonyadi, M. R., & Michalewicz, Z. (2015). Analysis of stability, local convergence, and transformation sensitivity of a variant of the particle swarm optimization algorithm. IEEE transactions on evolutionary computation, 20(3), 370-385.