RLEPSO:Reinforcement learning based Ensemble particle swarm optimizer

Authors: Yin Shiyuan

Date: 2021-06-29T00:00:00+00:00

Abstract

Evolution serves as the fundamental driving force behind the emergence of biological intelligence, while learning propels the advancement of human civilization. The synergy of evolution and learning collectively shapes the natural world.

Reinforcement learning has demonstrated remarkable efficacy across numerous domains. However, current research in optimization algorithms predominantly concentrates on evolution strategies, with limited investigation into learning mechanisms. Drawing inspiration from this observation, this paper introduces a novel particle swarm optimization algorithm, Reinforcement Learning based Ensemble Particle Swarm Optimizer (RLEPSO), that integrates reinforcement learning. The algorithm employs reinforcement learning for pre-training during the design phase to automatically discover more effective parameter configurations, thereby enhancing algorithmic performance and accelerating optimization processes. Furthermore, the algorithm incorporates two robust particle swarm variants and establishes weight parameters for different algorithms to better accommodate the solution requirements of diverse optimization problems, substantially improving algorithmic robustness. RLEPSO constructs multiple subswarms to increase the probability of locating the global optimum and enhance swarm diversity. The proposed RLEPSO is assessed on the CEC2013 benchmark set comprising 28 optimization test functions and compared against eight other particle swarm optimization variants, including three state-of-the-art algorithms. Experimental results demonstrate that RLEPSO achieves superior performance, surpassing all compared algorithms.

Full Text

Preamble

RLEPSO: Reinforcement Learning Based Ensemble Particle Swarm Optimizer

Evolution serves as the driving force behind the development of biological intelligence, while learning propels human civilization forward. The synergy between evolution and learning shapes the entire natural world. Although reinforcement learning has demonstrated remarkable effectiveness across numerous domains, researchers in the field of optimization algorithms have predominantly focused on evolutionary strategies, with learning-based approaches receiving comparatively little attention. Inspired by this gap, we propose a novel particle swarm optimization algorithm called Reinforcement Learning based Ensemble Particle Swarm Optimizer (RLEPSO) that integrates reinforcement learning into its core design. The algorithm employs reinforcement learning during a pre-training phase to automatically discover more effective parameter combinations, enabling superior performance and faster completion of optimization tasks. Furthermore, RLEPSO integrates two robust particle swarm variants and establishes weight parameters for different algorithms to better accommodate the solution requirements of diverse optimization problems, thereby significantly enhancing algorithmic robustness. By creating multiple sub-swarms, RLEPSO increases the probability of locating the global optimum while promoting particle swarm diversity. We evaluate the proposed RLEPSO on the CEC2013 optimization test function benchmark set comprising 28 functions and compare it against eight other particle swarm optimization variants, including three state-of-the-art algorithms. Experimental results demonstrate that RLEPSO achieves superior performance, outperforming all compared algorithms.

CCS Concepts: • Theory of computation → Reinforcement learning; Optimization with randomized search heuristics.

Additional Key Words and Phrases: Particle Swarm Optimization, Reinforcement Learning, Policy Gradient.

ACM Reference Format: Shiyuan Yin, Yi Liu, GuoLiang Gong, Huaxiang Lu, and Wenchang Li. 2021. RLEPSO: Reinforcement Learning Based Ensemble Particle Swarm Optimizer. 1, 1 (June 2021), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Optimization problems pervade all areas of science and engineering. As real-world optimization problems grow increasingly complex, obtaining optimal solutions through exhaustive search methods becomes generally infeasible. Consequently, heuristic algorithms are typically employed to obtain high-quality solutions. Currently, heuristic algorithms are advancing rapidly, with numer-

ous excellent methods emerging, including the Artificial Bee Colony algorithm (ABC) [?], Evolutionary Strategy (ES) [?], Differential Evolution (DE) [?], Evolutionary Programming (EP) [?], Genetic Algorithm (GA) [?], Ant Colony Optimization (ACO) [?], and Particle Swarm Optimization (PSO), among others. The performance of these algorithms has been validated on real-parameter optimization benchmark problems \cite{1, 9–11, 13, 17, 25}.

Among these algorithms, particle swarm optimization has garnered widespread attention due to its simple parameter configuration and rapid convergence speed. The particle swarm optimization algorithm is now extensively applied across various domains, including neuro-fuzzy system machine learning [?], scheduling problems [?, ?], autonomous navigation [?], system identification and control [?, ?], design optimization [?], and numerous other applications.

However, the particle swarm optimization algorithm suffers from several limitations, including premature convergence and prolonged search processes in later stages. Consequently, many researchers have proposed numerous improvements to address these shortcomings. In the Fitness-Distance-Ratio based Particle Swarm Optimizer (FDR-PSO), particles move toward nearby particles with higher fitness rather than simply moving toward the global best position [?]. In the Comprehensive Learning Particle Swarm Optimizer (CLPSO), each particle utilizes the historical best information of all other particles to update its own velocity [?]. In Orthogonal Learning Particle Swarm Optimization (OLPSO), the exemplar for each particle is generated using an orthogonal learning strategy combined with the swarm' s best experience [?]. In Locally Informed Particle Swarm Optimization (LIPS), particles follow local bests (the best experiences of neighboring particles) instead of the global best experience to locate the optimum across the search space [?]. In Self-Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients (HPSO-TVAC), each particle employs cognitive and social components to update its velocity, and when stagnation occurs in the search space, particles are reinitialized [?]. In Heterogeneous Particle Swarm Optimization (HPSO), a pool of different search behaviors exists, with each particle randomly selecting a behavior from the pool to update its velocity [?]. In the Ensemble Particle Swarm Optimizer [?], five PSO variants are integrated, dividing particles into two groups: the larger group updates itself using a PSO variant based on success rate, while the smaller group employs CLPSO for particle updates.

Although these algorithms have improved particle swarm performance in various ways, their manually set running parameters introduce redundancy. Currently, reinforcement learning is advancing rapidly, enabling the discovery of superior strategies through environmental interaction, which aligns well with optimization algorithms. Therefore, this paper proposes a Reinforcement Learning-based Ensemble Particle Swarm Optimizer (RLEPSO) that combines reinforcement learning with EPSO while incorporating additional improvements.

The performance of the proposed RLEPSO is tested on the CEC2013 test function set [?] comprising 28 functions. Compared against eight other particle

swarm algorithm variants, the results demonstrate that our algorithm significantly surpasses existing PSO variant algorithms in both exploration and optimization capabilities.

2.1 General Idea

EPSO has achieved promising results on optimization problems. However, as PSO variants become increasingly complex, they require more running parameters, making manual setting both cumbersome and suboptimal for achieving peak performance. Therefore, learning-based parameter optimization represents a superior approach. Moreover, since this parameter setting problem is essentially an optimization task that is non-differentiable and lacks a suitable loss function, traditional deep learning algorithms cannot be directly applied. Consequently, this paper employs the reinforcement learning algorithm DDPG (Deep Deterministic Policy Gradient) to obtain an effective action network that generates running parameters. This action network guides particle movement during the optimization process according to their current state. The complete RLEPSO algorithm comprises two components: optimization rules and action network A. To enhance the algorithm' s adaptability to different optimization objectives, we implement two algorithms—CLPSO and FDR-PSO—within RLEPSO.

2.2 Algorithm Details

This section briefly introduces CLPSO and FDR-PSO and details the operational mechanics of RLEPSO and the training of its action network. We assume readers possess a fundamental understanding of PSO. In the following content, *pbest* denotes a particle' s personal best experience and *gbest* represents the swarm' s best experience.

2.2.1 PSO Variants Employed in EPSO

Comprehensive Learning Particle Swarm Optimizer (CLPSO). In CLPSO, a particle learns from different particles' *pbest* values for different dimensions. The velocity of the i-th particle is updated using the following equation:

$$V_i^d = wV_i^d + c \cdot rand_d^i \cdot (pbest_{f_i(d)}^d - X_i^d)$$

In this equation, $f_i(d) = [f_i(1), f_i(2), ..., f_i(D)]$ determines which particle' s *pbest* the i-th particle should follow. CLPSO employs a P_c value to determine the target for each particle to follow. The target can be either the particle' s own *pbest* or another particle' s *pbest* for each dimension d. Every particle has its own P_{ci} value, generated by the following equation:

$$P_{ci} = a + b \cdot \frac{\exp(\frac{10(i-1)}{ps-1}) - 1}{\exp(10) - 1}$$

In this equation, ps is the population size, $a = 0.05$, and $b = 0.45$. When updating velocity for a dimension, a random value in $[0, 1]$ is generated and compared with P_{ci}. If the random value exceeds P_{ci}, the particle follows its own *pbest* for that dimension; otherwise, it follows another particle' s *pbest*. CLPSO uses tournament selection to choose a target particle. To avoid wasting function evaluations in unproductive directions, CLPSO defines a refreshing gap m representing a certain number of evaluations. During the period when a particle follows a target particle, the number of times the particle fails to improve is recorded as $flag_{clpso}$. If $flag_{clpso}$ exceeds m, the particle acquires a new target particle.

To integrate CLPSO with our RLEPSO, we define V_{CLPSO} using the following equation:

$$(V_{CLPSO})_i^d = rand_d^i \cdot (pbest_{f_i(d)}^d - X_i^d)$$

Fitness-Distance-Ratio Based Particle Swarm Optimization (FDR-PSO). Particles in FDR-PSO learn from their neighboring particle's experience (*nbest*) through a social learning component. Two criteria guide the selection of an appropriate target particle: (1) the target particle must be near the particle being updated, and (2) the target particle must be better than the particle being updated. To evaluate whether target particles meet these requirements, the Fitness-Distance-Ratio (FDR) is proposed as the ratio of fitness difference to one-dimensional distance. For the j-th particle, FDR is calculated using the following equation:

$$FDR = \frac{Fitness(X_i) - Fitness(X_j)}{|X_i - X_j|}$$

In this equation, X_i denotes the particle being updated and X_j denotes the target particle. In minimization problems, the particle that minimizes FDR is selected as the target particle. Once the target particle is selected, the i-th particle' s velocity is updated using the following equation:

$$V_i^d = w \cdot V_i^d + c_1 \cdot rand_1^i \cdot (pbest_i^d - X_i^d) + c_2 \cdot rand_2^i \cdot (gbest^d - X_i^d) + c_3 \cdot (nbest_j^d - X_i^d)$$

In this equation, $c_1 = 1$, $c_2 = 2$, and $c_3 = 2$. The *nbest* is the particle X_j identified by FDR. To integrate FDR-PSO with our RLEPSO, we define V_{FDR} using the following equation:

$$(V_{FDR})_i^d = rand_d^i \cdot (nbest_j^d - X_i^d)$$

2.2.2 RLEPSO Algorithm Optimization Process

EPSO employs a self-adaptive selection strategy to choose better PSO variants for solving optimization problems. To enable reinforcement learning and accelerate the optimization algorithm in RLEPSO, we remove the self-adaptive selection strategy and utilize a combined velocity update equation as follows:

$$V_{t+1} = w \cdot V_t + c_1 \cdot V_{CLPSO} + c_2 \cdot V_{FDR} + c_3 \cdot r_1 \cdot (gbest - X) + c_4 \cdot r_2 \cdot (pbest - X)$$

In this equation, V_{CLPSO} and V_{FDR} are as introduced in the previous section. The *pbest* represents the particle's personal best experience, and *gbest* represents the swarm's best experience. Based on the current running state, the coefficients w, c_1, c_2, c_3, and c_4 are generated by the actor network. The variables r_1 and r_2 are uniformly distributed random numbers between 0 and 1. To enhance particle diversity and increase the probability of finding the global optimum, all particles in RLEPSO are divided into five sub-swarms. Each sub-swarm possesses its own coefficients (w, c_1, etc.) but shares the same *gbest*.

To prevent particles from becoming trapped in local optima, a mutation stage follows velocity updating. During this stage, a random number r_5 between 0 and 1 is first generated and then compared with $C_{mutation} \cdot 0.01 \cdot flag_{clpso}$. If r_5 is smaller than this threshold, mutation occurs and the particle position is reinitialized within the solution space.

At the end of each period, particles move according to their velocity, after which particle fitness and historical best experiences are updated.

2.2.3 Actor Network

In this paper, the action network is designed as a compact network with 1-dimensional input and 35-dimensional output, incurring minimal additional computational cost. In each round of the particle swarm, the input content is recorded as the state vector S_t, and the output content is recorded as the action vector A_t. All action values range between 0 and 1.

The state vector generation method is as follows:

$$S_t = fe_t / fe_{max}$$

where fe_t represents the number of function evaluations executed in round t, and fe_{max} specifies the total number of function evaluations for the optimization process. The obtained action vector A_t is 35-dimensional, divided into five

groups targeting each sub-swarm. For a sub-swarm, the action vector is 7-dimensional from $a[0]$ to $a[6]$. The parameters w, c_1, c_2, c_3, c_4, and $c_{mutation}$ required for each optimization round are generated based on $a[0]$ to $a[6]$ using the following formulas:

$$c_{mutation} = a[0] \cdot 0.01 \cdot flag_{clpso}$$

$$w = a[1] \cdot 0.8 + 0.1$$

$$scale = \frac{1}{a[3] + a[4] + a[5] + a[6] + 0.00001} \cdot a[2] \cdot 8$$

$$c_1 = scale \cdot a[3]$$

$$c_2 = scale \cdot a[4]$$

$$c_3 = scale \cdot a[5]$$

$$c_4 = scale \cdot a[6]$$

The *scale* parameter controls the range of velocity changes and prevents unstable particle oscillation.

2.2.4 Training Algorithm: DDPG

To train the actor network , we employ a reinforcement learning algorithm called Deep Deterministic Policy Gradient (DDPG) [**?**]. We briefly introduce DDPG in the following section.

In a standard reinforcement learning environment, each agent interacts with the environment, aiming to maximize environmental rewards. This interactive process is formally described as a Markov Decision Process (MDP), represented by the four-tuple (S, A, R, P). Here, S is the state space, A is the action space, $R : S \times A \to \mathbb{R}$ is the reward function, and $P : S \times A \times S \to [0, 1]$ is the transition probability. In this environment, an agent learns a policy $\pi : S \to A$ to maximize environmental rewards.

The action value function Q is typically used to represent the reward for performing an action in state s:

$$Q(s_t, a_t) = \mathbb{E}[R_t|s = s_t, a = a_t] = \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i)\right]$$

In this equation, $r(s_i, a_i)$ represents the immediate reward from the current action, while $Q(s_t, a_t)$ represents the long-term reward from the current action.

DDPG designs two deep neural networks: the action value network $Q(s_t, a_t|\theta_Q)$ and the action network $\mu(s_t|\theta_\mu)$, where θ_Q and θ_μ are network parameters. The action network maps the state space to the action space, directly generating required actions based on the state (effectively implementing policy π). The action value network approximates the action value function Q and provides gradients for training the action network.

Training the action value network involves minimizing the loss function:

$$L(\theta_Q) = (r(s_t, a_t) + \gamma Q'(s_{t+1}, a_{t+1}|\theta_Q) - Q(s_t, a_t|\theta_Q))^2$$

where Q' is the target value network whose weights are synchronized from network Q. Updating the action network parameters requires the policy gradient algorithm, with the gradient update direction as follows:

$$\nabla_{\theta_\mu} Q(s, a|\theta_Q)|_{s=s_t, a=\mu(s_t,v)} = \nabla_a Q(s, a|\theta_Q)|_{s=s_t, a=\mu(s_t,v)} \nabla_{\theta_\mu} \mu(s|\theta_\mu)|_{s=s_t}$$

Through iteration, we ultimately obtain an effective action network μ.

Training Details. In the training process, each episode represents a complete optimization process of RLEPSO for an objective function, while each epoch represents one round of RLEPSO during the optimization of the objective function. Performing an action in the environment involves inputting the generated action vector into RLEPSO, optimizing the target for one round, and obtaining the reward and next state for the subsequent round.

The algorithm' s reward value is set as follows: when the best value of PSO in the current environment changes, the reward is $+1$; otherwise, the reward is -1. This setting aims to improve RLEPSO' s optimization speed. The test function used to initialize the environment during each training episode is randomly selected from CEC2013. The pseudocode is presented in Algorithm 2.

3 Experiments and Results

This paper evaluates the proposed RLEPSO algorithm' s performance using the shifted and rotated CEC2013 benchmark functions. The CEC2013 benchmark suite consists of 28 diverse functions, including unimodal, multimodal, expanded, and hybrid composition functions.

The comparison algorithms include various PSO variants: Inertia Weight PSO (PSO), Comprehensive Learning PSO (CLPSO), Self-Organizing Hierarchical PSO with Time-Varying Acceleration Coefficients (HPSO-TVAC), Fitness-Distance-Ratio based PSO (FDR-PSO), Distance-Based Locally Informed PSO (LIPS), Orthogonal Learning PSO (OLPSO), Static Heterogeneous Swarm Optimization (sHPSO), and Ensemble Particle Swarm Optimizer (EPSO). These algorithms were introduced in the previous section. The first five algorithms represent popular developments, while the latter three are state-of-the-art algorithms in the particle swarm variant domain. The settings for these algorithms follow those in [**?**].

All experiments were conducted ten times independently, with the average value used as the final result. To test the optimization algorithm' s adaptability under different conditions, the evaluation includes three different dimensions: 50, 30, and 10.

presents the final solutions obtained for each function at dimension 50. In this table, first-place results are marked in bold and second-place results in blue. RLEPSO ranks among the top four in all functions, achieving first-place results in 11 functions and second-place results in 10 functions. The worst ranking for RLEPSO is fourth place. This demonstrates that RLEPSO possesses strong optimization capabilities and consistently ranks among the best across various evaluation functions. Moreover, it shows that RLEPSO is highly stable, capable of adapting to diverse complex optimization objectives, and can obtain more reliable solutions.

shows the final average rankings of different algorithms across each dimension. It is evident that RLEPSO leads by an absolute advantage at dimensions 50, 30, and 10. At dimension 50, the average ranking surpasses the second-place algorithm by 0.68; at dimension 30, by 0.61; and at dimension 10, by 0.43. In the comprehensive average ranking across all dimensions, RLEPSO is 0.6 ahead of the second-place algorithm, fully reflecting its superiority in solution accuracy.

However, it is also observable that since the training process was performed at dimension 50, the lead diminishes somewhat in the experiments at dimensions 30 and 10, indicating that this training method exhibits certain problem-specific relevance. In practical applications, a single deployment of an algorithm typically addresses a specific problem. Therefore, pre-training for a specific problem and subsequently deploying the algorithm is both feasible and highly effective.

In summary, RLEPSO achieves higher optimization accuracy than comparable algorithms and can effectively solve high-dimensional complex numerical optimization problems. The swarm optimization algorithm based on reinforcement learning demonstrates superior optimization and exploration capabilities compared to swarm optimization algorithms with manually set parameters.

4 Conclusion

This paper proposes an ensemble particle swarm optimization algorithm based on reinforcement learning that integrates two particle swarm algorithm variants —CLPSO and FDR-PSO. The algorithm utilizes the reinforcement learning algorithm DDPG to train an action network A, which efficiently provides running parameters according to the current optimization stage. Additionally, to enhance the algorithm' s global search capabilities, all particles are divided into five sub-swarms, with each sub-swarm independently receiving running parameters from actor network A. Moreover, the algorithm introduces a new mutation step during the movement process to prevent premature entrapment in local optima. During this stage, particles are randomly reinitialized based on the algorithm' s current configuration and the number of times particles cease improving. This approach yields excellent population diversity and global search capabilities for RLEPSO.

The RLEPSO algorithm' s performance is validated through the CEC2013 standard test set and compared against eight other particle swarm variants. The results show that RLEPSO performs well on all test functions, achieving top-two results on 21 out of 28 functions. RLEPSO outperforms both its individual PSO variants and recent state-of-the-art PSO algorithms. Future research directions for RLEPSO include incorporating additional optimization algorithms into the framework and applying DDPG to other optimization algorithms.

REFERENCES

[1] Swagatam Das and Ponnuthurai N Suganthan. 2010. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Nanyang Technological University, Kolkata (2010), 341–359.

[2] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. IEEE computational intelligence magazine 1, 4 (2006), 28–39.

[3] Andries P. Engelbrecht. 2010. Heterogeneous particle swarm optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6234 LNCS (2010), 191–202. https://doi.org/10.1007/978-3-642-15461-4_{17}

[4] Vitaliy Feoktistov. 2006. Differential evolution. Springer.

[5] Bo He, Lulu Ying, Shujing Zhang, Xiao Feng, Tianhong Yan, Rui Nian, and Yue Shen. 2015. Autonomous navigation based on unscented-FastSLAM using particle swarm optimization for autonomous underwater vehicles. Measurement 71 (2015), 89–101.

[6] Wenbin Hu, Huan Wang, Zhenyu Qiu, Cong Nie, and Liping Yan. 2018. A quantum particle swarm optimization driven urban traffic light scheduling model. Neural Computing and Applications 29, 3 (2018), 901–911.

[7] Dervis Karaboga. 2010. Artificial bee colony algorithm. scholarpedia 5, 3 (2010), 6915.

[8] TH Kim, I Maruta, and T Sugie. 2010. A simple and efficient constrained particle swarm optimization and its application to engineering design problems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 224, 2 (2010), 389–400.

[9] JJ Liang, BY Qu, PN Suganthan, and Q Chen. 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 29 (2014), 625–640.

[10] JJ Liang, BY Qu, PN Suganthan, and Alfredo G Hernández-Díaz. 2013. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212, 34 (2013), 281–295.

[11] JJ Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. 2006. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. Journal of Applied Mechanics 41, 8 (2006), 8–31.

[12] Jing J. Liang, A. K. Qin, Ponnuthurai Nagaratnam Suganthan, and S. Baskar. 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation 10, 3 (2006), 281–295. https://doi.org/10.1109/TEVC.2005.857610

[13] Jing J Liang, Bo Y Qu, and Ponnuthurai N Suganthan. 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635 (2013), 490.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings (2016). arXiv:1509.02971

[15] Jianshan Lu. 2017. System identification of force transducers for dynamic measurements using particle swarm optimization. Journal of Vibroengi- neering 19, 2 (2017), 864–877.

[16] Nandar Lynn and Ponnuthurai Nagaratnam Suganthan. 2017. Ensemble particle swarm optimizer. Applied Soft Computing Journal 55 (2017), 533–548.

https://doi.org/10.1016/j.asoc.2017.02.007

[17] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. 2010. Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. Nanyang Technological University, Singapore 24 (2010).

[18] Seyedali Mirjalili. 2019. Genetic algorithm. In Evolutionary algorithms and neural networks. Springer, 43–55.

[19] Maroua Nouiri, Abdelghani Bekrar, Abderezak Jemai, Smail Niar, and Ahmed Chiheb Ammari. 2018. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. Journal of Intelligent Manufacturing 29, 3 (2018), 603–615.

[20] T. Peram, K. Veeramachaneni, and C. K. Mohan. 2003. Fitness-distance-ratio based particle swarm optimization. 2003 IEEE Swarm Intelligence Symposium, SIS 2003 - Proceedings 2 (2003), 174–181. https://doi.org/10.1109/SIS.2003.1202264

[21] Joram Piatigorsky and Graeme J Wistow. 1989. Enzyme/crystallins: gene sharing as an evolutionary strategy. Cell 57, 2 (1989), 197–199.

[22] B. Y. Qu, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. 2013. A distance-based locally informed particle swarm model for multimodal optimization. IEEE Transactions on Evolutionary Computation 17, 3 (2013), 387–402. https://doi.org/10.1109/TEVC.2012.2203138

[23] Asanga Ratnaweera, Saman K. Halgamuge, and Harry C. Watson. 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation 8, 3 (2004), 240–255. https://doi.org/10.1109/TEVC.2004.826071

[24] KV Shihabudheen, M Mahesh, and GN Pillai. 2018. Particle swarm optimization based extreme learning neuro-fuzzy system for regression and classification. Expert Systems with Applications 92 (2018), 474–484.

[25] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL report 2005005, 2005 (2005), 2005.

[26] Xiaoping Yang, Xueying Chen, Riting Xia, and Zhihong Qian. 2018. Wireless sensor network congestion control based on standard particle swarm optimization and single neuron PID. Sensors 18, 4 (2018), 1265.

[27] Xin Yao, Yong Liu, and Guangming Lin. 1999. Evolutionary programming made faster. IEEE Transactions on Evolutionary computation 3, 2 (1999),

[28] Zhi Hui Zhan, Jun Zhang, Yun Li, and Yu Hui Shi. 2011. Orthogonal learning particle swarm optimization. IEEE Transactions on Evolutionary Computation 15, 6 (2011), 832–847. https://doi.org/10.1109/TEVC.2010.2052054