Gaussian-Valued Particle Swarm Optimization

Kyle Robert Harrison[1], Beatrice M. Ombuki-Berman[2], and Andries P. Engelbrecht[1]

[1] Department of Computer Science
University of Pretoria, Pretoria, South Africa
kharrison@outlook.com, engel@cs.up.ac.za
[2] Department of Computer Science
Brock University, St. Catharines, Canada
bombuki@brocku.ca

Abstract. This paper examines the position update equation of the particle swarm optimization (PSO) algorithm, leading to the proposal of a simplified position update based upon a Gaussian distribution. The proposed algorithm, Gaussian-valued particle swarm optimization (GVPSO), generates probabilistic positions by retaining key elements of the the canonical update procedure while also removing the need to specify values for the traditional PSO control parameters. Experimental results across a set of 60 benchmark problems indicate that GVPSO outperforms both the standard PSO and the bare bones particle swarm optimization (BBPSO) algorithm, which also employs a Gaussian distribution to generate particle positions.

1 Introduction

The particle swarm optimization (PSO) algorithm [22] is a stochastic optimization technique based upon the social dynamics of a flock of birds. The PSO algorithm generates new positions stochastically based upon the position of two key attractors in the search space, namely the personal and neighbourhood best positions. The step sizes, and therefore the degree of exploration and exploitation, are then controlled via the values of three control parameters [1, 4, 20, 28]. The values of the control parameters directly influence the particle movement patterns [1, 3]. However, the best control parameter values are problem dependent and effective tuning is needed to improve PSO performance [2, 4, 37].

While parameter tuning is clearly warranted in the PSO algorithm, it is typically a time-consuming process whereby a large number of candidate parameter configurations must be analysed. Fortunately, there have been a number of studies that have suggested general-purpose PSO parameters based on empirical evidence [3–5, 8, 18, 19, 28, 37, 41]. While these studies have made use of the implicit assumption that *a priori* tuning of control parameters is sufficient to optimize performance, recent evidence suggests that the best PSO parameters to employ change over time [18]. Similar results have also been found for heterogeneous PSOs [26, 29, 30, 38] and for dynamic PSOs [24].

An alternative approach is to use PSO variants that do not rely on *a priori* control parameter values. A prominent example of this approach lies in the development of self-adaptive PSO (SAPSO) techniques, which continuously adapt the values of their control parameters throughout the search process [25, 27, 31, 32, 34–36, 39, 40]. However, many of the SAPSO algorithms have been shown to exhibit either premature convergence or rapid divergence, thereby leading to poor performance [14, 15, 17, 42]. An additional example is the bare bones PSO (BBPSO) [21], which updates particle positions probabilistically using a Gaussian distribution. However, the manner in which particle positions are created via BBPSO is strikingly dissimilar to how the conventional PSO determines updated particle positions.

In this paper, a new PSO variant is proposed by formulating a new probabilistic approach to generating particle positions. The new approach is inspired by the BBPSO algorithm, but differs significantly in the manner by which particle positions are generated. Notably, the proposed algorithm generates particle positions using a model that more closely resembles the canonical PSO, which as this paper will demonstrate, provides a clear performance advantage over BBPSO and other PSO configurations.

The remainder of this paper is structured as follows. Section 2 provides the necessary background information about PSO and BBPSO. The proposed algorithm is described in Section 3, while Section 4 details the empirical analysis and results. Finally, concluding remarks and avenues of future work are provided in Section 5.

2 Background

This section provides the necessary background information about the PSO and BBPSO algorithms. The PSO algorithm is described in Section 2.1 while the BBPSO algorithm is outlined in Section 2.2.

2.1 Particle Swarm Optimization

The PSO algorithm [22] consists of a collection of agents, referred to as particles, which each represent a candidate solution to an optimization problem. Each particle retains three pieces of information, namely its current position, velocity, and its (personal) best position found within the search space. Particle positions are updated each iteration via the calculation and subsequent addition of a velocity to the particle's current position. A particle's velocity is based on its attraction towards two (promising) locations in the search space, namely the best position found by the particle itself and the best position found by any particle within the particle's immediate neighbourhood [23]. The neighbourhood of a particle is defined as the other particles within the swarm from which it may take influence, which is most commonly the entire swarm or, alternatively, the immediate neighbours when arranged in a ring [23].

To facilitate movement in the PSO algorithm, the velocity is calculated for particle i according to the inertia weight model [34] as

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2ij}(t)(\hat{y}_{ij}(t) - x_{ij}(t)), \quad (1)$$

where $v_{ij}(t)$ and $x_{ij}(t)$ are the velocity and position in dimension j at time t, respectively. The inertia weight is given by ω while c_1 and c_2 represent the cognitive and social acceleration coefficients, respectively. The stochastic component of the algorithm is provided by the random values $r_{1ij}(t), r_{2ij}(t) \sim U(0,1)$, which are independently sampled each iteration for all components of each particle's velocity. Finally, $y_{ij}(t)$ and $\hat{y}_{ij}(t)$ denote the personal and neighbourhood best positions in dimension j, respectively. Particle positions are then updated according to

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \quad (2)$$

2.2 Barebones Particle Swarm Optimization

When examining particle movement patterns, Kennedy noted that particle positions formed a bell curve centred around the midpoint between the global and personal best positions [21]. Based on this result, the BBPSO algorithm [21] eliminates the velocity component of PSO and rather updates particle positions probabilistically according to

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < e \\ \mathcal{N}\left(\frac{c_1 y_{ij}(t) + c_2 \hat{y}_{ij}(t)}{c_1 + c_2}, |y_{ij}(t) - \hat{y}_{ij}(t)|\right) & \text{otherwise} \end{cases}, \quad (3)$$

where $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ and e is a parameter representing the per-dimension chance of selecting the personal best position. In the original formulation of BBPSO, the control parameters were set as $c_1 = c_2 = 1$ [21]. Later theoretical results supported the observation of Kennedy by showing that, using the stagnation and deterministic assumptions, each particle will converge to the point $\frac{c_1 \boldsymbol{y}_i + c_2 \hat{\boldsymbol{y}}_i}{c_1 + c_2}$ [2, 37].

3 Gaussian Valued Particle Swarm Optimization

To provide the motivation for the proposed algorithm, consider the PSO velocity equation given in Eq. (1) when $\omega = 0$. With no inertia, the velocity calculation simplifies to

$$v_{ij}(t+1) = c_1 r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2ij}(t)(\hat{y}_{ij}(t) - x_{ij}(t)). \quad (4)$$

Note that because $r_{1ij}(t), r_{2ij}(t) \sim U(0,1)$, Eq. (4) can be reformulated as

$$v_{ij}(t+1) = v_{1ij}(t) + v_{2ij}(t) \quad (5)$$

where $v_{1ij}(t) \sim U(0, c_1(y_{ij}(t) - x_{ij}(t)))$ and $v_{2ij}(t) \sim U(0, c_2(\hat{y}_{ij}(t) - x_{ij}(t)))$[3]. It can be easily observed that the position update becomes a sum of two uniform distributions, thereby leading to a trapezoidal distribution. The shape of the resulting trapezoidal distribution is then governed by the distance between the current particle's position and the position of the personal and neighbourhood best, respectively. Even with the reintroduction of the inertia component, the same general observation can be made; the particle position update depends heavily upon not only the personal and neighbourhood best positions, but rather the distance between the current particle and these two attractors.

The position update mechanism for GVPSO is formulated by employing a Gaussian distribution centred at a random point taken from the aforementioned trapezoidal distribution. The Gaussian distribution is used to modulate the particle step sizes based upon the distance between the current position and the personal and neighbourhood best positions. Specifically, an ancillary position, $\Delta_{ij}(t)$, is calculated for each particle in every dimension using Eqs. (1) and (2) with $\omega = 0$ and $c_1 = c_2 = 1$. This effectively retains the core movement pattern of PSO without the reliance on control parameter values. The particle's new position is then determined using a Gaussian distribution centred between the current position and $\Delta_{ij}(t)$ with a standard deviation based on the magnitude of the distance between the current position and $\Delta_{ij}(t)$ according to

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < e \\ \mathcal{N}\left(\frac{x_{ij}(t)+\Delta_{ij}(t)}{2}, |\Delta_{ij}(t) - x_{ij}(t)|\right) & \text{otherwise} \end{cases}, \qquad (6)$$

where e is the exploitation parameter, as seen in Eq. (3). Note that GVPSO, in the same manner as BBPSO, eliminates the need for the conventional PSO parameters ω, c_1, and c_2. However, the GVPSO algorithm differs from BBPSO by creating particle positions that more closely mimic the canonical position update of PSO through the use of distance information and thus the two attractors remain to have a strong influence. Furthermore, the step sizes in the GVPSO are implicitly controlled by the distances between the current particle and the two attractors, thereby leading to diminishing step sizes as the positions and attractors inevitably converge. Thus, the GVPSO is expected to exhibit both initial exploration and exploitation in the later phase of the search process.

4 Experimental Results and Discussion

This section presents the experimental design regarding the empirical examination of GVPSO. Section 4.1 describes the parameterization, benchmark suite, and statistical analysis. Section 4.2 presents a sensitivity analysis on the exploitation parameter while Section 4.3 presents a comparison of GVPSO to other PSO variants.

[3] Without loss of generality, this assumes that $c_1(y_{ij}(t) - x_{ij}(t)) > 0$ and $c_2(\hat{y}_{ij}(t) - x_{ij}(t)) > 0$, otherwise the bounds must be flipped, i.e., 0 becomes the upper bound.

4.1 Experimental Setup

To first examine the effect of the exploitation probability parameter e, 10 values of e were examined for GVPSO and BBPSO, namely values between 0.0 and 0.9 in increments of 0.1. Linearly decreasing variants (GVPSO-LD and BBPSO-LD), whereby the value of e was linearly decreased from 0.9 to 0.0, were also examined. The performance of GVPSO was then compared against the following PSO strategies:

- BBPSO
- Three static PSO parameter configurations: PSO-1 ($\omega = 0.7298, c_1 = c_2 = 1.49618$) [7], PSO-2 ($\omega = 0.729, c_1 = 2.0412, c_2 = 0.9477$) [4], and PSO-7 ($\omega = 0.785, c_1 = c_2 = 1.331$) [41], which were found to be the best performing of 14 commonly recommended PSO parametrizations [16]
- PSO with time-varying acceleration coefficients (PSO-TVAC) [32]
- PSO with improved random constants (PSO-iRC) [16]

All examined variants consisted of 30 particles arranged in a star neighbourhood and used a synchronous update strategy. To prevent invalid attractors, a particle's personal best position was only updated if a new position had a better objective function value and was within the feasible bounds of the search space. For the BBPSO algorithm, the original parametrization of $c_1 = c_2 = 1$ was used. Where applicable, particle velocities were initialized to zero [9]. For PSO-TVAC, the social acceleration coefficient was linearly increased from 0.5 to 2.5 while the values of the cognitive and inertia control parameters were linearly decreased from 2.5 to 0.5 and 0.9 to 0.4, respectively. For PSO-iRC, parameter configurations were re-sampled every 5 iterations (i.e., according to PSO-iRC-p5 [16]). The value of the objective function (i.e., the fitness), averaged over 50 independent runs each consisting of 5000 iterations, was taken as the measure of performance for each algorithm.

Benchmark Problems A suite of 60 minimization problems, originally used by [10], were used in this study. The suite has been demonstrated to include a range of different landscape characteristics [13]. All functions were optimized in 30 dimensions. Further information about the benchmark suite can be found in [10] and [14].

Statistical Analysis Statistical analysis of results was done by way of Friedman's test for multiple comparisons among all methods [11, 12], as recommended by Derrac *et al.* [6]. Furthermore, Shaffer's post-hoc procedure [33] was performed as a means to identify the pairwise comparisons that produced significant differences. Finally, the statistical results are visualized via critical difference plots, whereby algorithms to the left of the plot (i.e., those with lower average ranks) demonstrated superior performance. The critical difference (CD) denotes the difference in average rank that was found to be statistically significant. Therefore, algorithms that are grouped by a line (i.e., those with a difference in rank less than CD) were found to have statistically insignificant differences in performance.

4.2 Examining the Exploitation Probability

Figures 1 and 2 show the critical difference plots for the examined values of e for both the GVPSO and BBPSO algorithms over the entire set of problems. While the exact values that lead to the best performance were different among the two algorithms, the general trends were the same. In general, mid-range values of e (i.e., 0.4–0.7) tended to perform the best, showing that both exploration and exploitation were beneficial to the GVPSO algorithm. Based upon these results, GVPSO and BBPSO with values of e set to 0.5, 0.6, and 0.7 were compared against other PSO techniques in the next section.

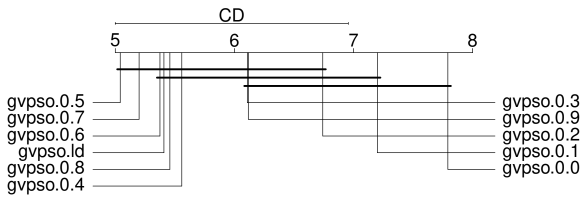

Fig. 1: Comparison of GVPSO exploit probabilities over all 60 benchmark problems.

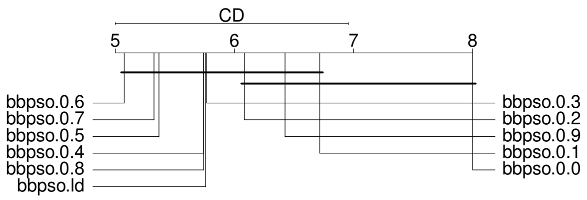

Fig. 2: Comparison of BBPSO exploit probabilities over all 60 benchmark problems.

4.3 Comparison with Other Particle Swarm Optimization Techniques

This section presents the results from comparing GVPSO (with $e = \{0.5, 0.6, 0.7\}$) against the other PSO variants. Fig. 3 shows the results across all benchmark problems. It was first observed that the best average ranks across all benchmark problems were attained by the three configurations of GVPSO, clearly indicating the merit of this approach. Despite the better average rank attained by GVPSO, the critical difference plot indicates there was no significant difference in performance between the different GVPSO and BBPSO configurations as well as PSO-2. However, it was also observed from Fig. 3 that PSO-2 attained a notably worse average rank than each of the GVPSO and BBPSO configurations. The remaining PSO variants, namely PSO-1, PSO-7, PSO-TVAC, and PSO-iRC-p5 all performed significantly worse than GVPSO.

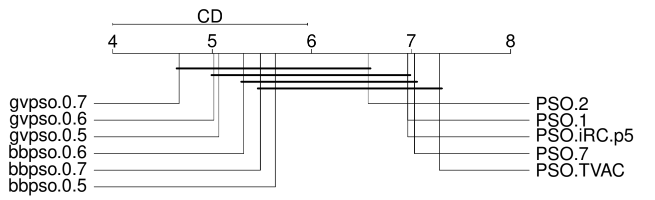

Fig. 3: Comparison of GVPSO with other PSO variants over all 60 benchmark problems.

5 Conclusions and Future Work

This paper proposed a new particle swarm optimization (PSO) variant, entitled Gaussian-valued PSO (GVPSO), which generates particle positions probabilistically according a Gaussian distribution. The GVPSO algorithm is loosely inspired by the bare bones PSO (BBPSO) but differs significantly from the BBPSO algorithm by generating particles according to a distribution that more closely resembles the conventional PSO position update. An analysis of the single parameter of GVPSO was first performed, followed by a comparison of GVPSO to BBPSO and five additional PSO configurations. Results indicate that GVPSO generally outperforms the other strategies.

An immediate avenue of future work lies in the self-adaptation of the single GVPSO parameter, resulting in a parameter-free algorithm. Further work will also examine the proposed algorithm in different dimensionalities and compare its performance against additional PSO variants, including improved implementations of BBPSO.

Acknowledgments. This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 46712). The opinions, findings and conclusions or recommendations expressed in this article is that of the author(s) alone, and not that of the NRF. The NRF accepts no liability whatsoever in this regard. This work is also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. van den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. thesis, University of Pretoria (2001)
2. van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information Sciences **176**(8), 937–971 (2006). https://doi.org/10.1016/j.ins.2005.02.003
3. Bonyadi, M., Michalewicz, Z.: Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. IEEE Transactions on Evolutionary Computation **21**(3), 1–1 (2016). https://doi.org/10.1109/TEVC.2016.2605668
4. Carlisle, A., Dozier, G.: An off-the-shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization. vol. 1, pp. 1–6. Purdue School of Engineering and Technology (2001)
5. Clerc, M.: Stagnation analysis in particle swarm optimisation or what happens when nothing happens. Tech. Rep. 1, HAL (2006)
6. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011). https://doi.org/10.1016/j.swevo.2011.02.002
7. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. vol. 12, pp. 39–43. IEEE (2008). https://doi.org/10.1109/MHS.1995.494215
8. Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation. pp. 84–88. IEEE (2000). https://doi.org/10.1109/CEC.2000.870279
9. Engelbrecht, A.: Particle swarm optimization: velocity initialization. In: Proceedings of the 2012 IEEE Congress on Evolutionary Computation. pp. 1–8. IEEE (2012). https://doi.org/10.1109/CEC.2012.6256112
10. Engelbrecht, A.: Particle swarm optimization: global best or local best? In: Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. pp. 124–135. IEEE (2013). https://doi.org/10.1109/BRICS-CCI-CBIC.2013.31
11. Friedman, M.: A comparison of alternative tests of significance for the problem of *m* rankings. Annals of Mathematical Statistics **11**(1), 86–92 (1940). https://doi.org/10.1214/aoms/1177731944
12. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of American Statistical Association **32**(32), 675–701 (1937). https://doi.org/10.1080/01621459.1937.10503522
13. Garden, R.W., Engelbrecht, A.P.: Analysis and classification of optimisation benchmark functions and benchmark suites. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation. pp. 1641–1649. IEEE (2014). https://doi.org/10.1109/CEC.2014.6900240
14. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: Inertia weight control strategies for particle swarm optimization. Swarm Intelligence **10**(4), 267–305 (2016). https://doi.org/10.1007/s11721-016-0128-z
15. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: The sad state of self-adaptive particle swarm optimizers. In: Proceedings of the 2016 IEEE Congress on Evolutionary Computation. pp. 431–439. IEEE (2016). https://doi.org/10.1109/CEC.2016.7743826

16. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: An adaptive particle swarm optimization algorithm based on optimal parameter regions. In: Proeceedings of the 2017 IEEE Symposium Series on Computational Intelligence. pp. 1606–1613. IEEE (2017). https://doi.org/10.1109/SSCI.2017.8285342
17. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: Self-adaptive particle swarm optimization: a review and analysis of convergence. Swarm Intelligence (2017). https://doi.org/10.1007/s11721-017-0150-9
18. Harrison, K.R., Engelbrecht, A.P., Ombuki-Berman, B.M.: Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm. Swarm and Evolutionary Computation (2018). https://doi.org/10.1016/j.swevo.2018.01.006
19. Jiang, M., Luo, Y., Yang, S.: Particle swarm optimization - stochastic trajectory analysis and parameter selection. In: Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, pp. 179–198. No. December, I-Tech Education and Publishing (2007). https://doi.org/10.5772/5104
20. Jiang, M., Luo, Y., Yang, S.: Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Information Processing Letters **102**(1), 8–16 (2007). https://doi.org/10.1016/j.ipl.2006.10.005
21. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. pp. 80–87. IEEE (2003). https://doi.org/10.1109/SIS.2003.1202251
22. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the International Conference on Neural Networks. vol. 4, pp. 1942–1948. IEEE (1995). https://doi.org/10.1109/ICNN.1995.488968
23. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation. vol. 2, pp. 1671–1676. IEEE (2002). https://doi.org/10.1109/CEC.2002.1004493
24. Leonard, B.J., Engelbrecht, A.P.: On the optimality of particle swarm parameters in dynamic environments. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation. pp. 1564–1569. IEEE (2013). https://doi.org/10.1109/CEC.2013.6557748
25. Leu, M.S., Yeh, M.F.: Grey particle swarm optimization. Applied Soft Computing **12**(9), 2985–2996 (2012). https://doi.org/10.1016/j.asoc.2012.04.030
26. Li, C., Yang, S., Nguyen, T.T.: A self-learning particle swarm optimizer for global optimization problems. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **42**(3), 627–646 (2012). https://doi.org/10.1109/TSMCB.2011.2171946
27. Li, X., Fu, H., Zhang, C.: A self-adaptive particle swarm optimization algorithm. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering. vol. 5, pp. 186–189. IEEE (2008). https://doi.org/10.1109/CSSE.2008.142
28. Liu, Q.: Order-2 stability analysis of particle swarm optimization. Evolutionary Computation **23**(2), 187–216 (2015). https://doi.org/10.1162/EVCO_a_00129
29. Montes de Oca, M.A., Peña, J., Stützle, T., Pinciroli, C., Dorigo, M.: Heterogeneous particle swarm optimizers. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation. pp. 698–705. IEEE (2009). https://doi.org/10.1109/CEC.2009.4983013
30. Nepomuceno, F.V., Engelbrecht, A.P.: A self-adaptive heterogeneous PSO for real-parameter optimization. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation. pp. 361–368. IEEE (2013). https://doi.org/10.1109/CEC.2013.6557592

31. Nickabadi, A., Ebadzadeh, M.M., Safabakhsh, R.: A novel particle swarm optimization algorithm with adaptive inertia weight. Applied Soft Computing **11**(4), 3658–3670 (2011). https://doi.org/10.1016/j.asoc.2011.01.037
32. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Transactions on Evolutionary Computation **8**(3), 240–255 (2004). https://doi.org/10.1109/TEVC.2004.826071
33. Shaffer, J.P.: Modified sequentially rejective multiple test procedures. Journal of the American Statistical Association **81**(395), 826–831 (1986). https://doi.org/10.1080/01621459.1986.10478341
34. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation. pp. 69–73. IEEE (1998). https://doi.org/10.1109/ICEC.1998.699146
35. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation. vol. 3, pp. 1945–1950. IEEE (1999). https://doi.org/10.1109/CEC.1999.785511
36. Tanweer, M., Suresh, S., Sundararajan, N.: Self regulating particle swarm optimization algorithm. Information Sciences **294**, 182–202 (2015). https://doi.org/10.1016/j.ins.2014.09.053
37. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters **85**(6), 317–325 (2003). https://doi.org/10.1016/S0020-0190(02)00447-7
38. Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-adaptive learning based particle swarm optimization. Information Sciences **181**(20), 4515–4538 (2011). https://doi.org/10.1016/j.ins.2010.07.013
39. Xu, G.: An adaptive parameter tuning of particle swarm optimization algorithm. Applied Mathematics and Computation **219**(9), 4560–4569 (2013). https://doi.org/10.1016/j.amc.2012.10.067
40. Yang, X., Yuan, J., Yuan, J., Mao, H.: A modified particle swarm optimizer with dynamic adaptation. Applied Mathematics and Computation **189**(2), 1205–1213 (2007). https://doi.org/10.1016/j.amc.2006.12.045
41. Zhang, W., Ma, D., Wei, J.j., Liang, H.f.: A parameter selection strategy for particle swarm optimization based on particle positions. Expert Systems with Applications **41**(7), 3576–3584 (2014). https://doi.org/10.1016/j.eswa.2013.10.061
42. van Zyl, E., Engelbrecht, A.: Comparison of self-adaptive particle swarm optimizers. In: Proceedings of the 2014 IEEE Symposium on Swarm Intelligence. pp. 1–9. IEEE (2014). https://doi.org/10.1109/SIS.2014.7011775