PSO–FWA: A New Hybrid Algorithm for Solving Nonlinear Equation Systems

Sérgio Ribeiro[1(✉)] and Luiz Guerreiro Lopes[2]

[1] Postgraduate Program in Informatics Engineering, University of Madeira, Funchal, Madeira Is., Portugal
sergioribeiro_91@hotmail.com

[2] Faculty of Exact Sciences and Engineering, University of Madeira, 9020-105 Funchal, Madeira Is., Portugal
lopes@uma.pt

Abstract. Nature-inspired optimization algorithms have been proposed for solving hard optimization problems, including the optimization-based solution of difficult systems of nonlinear equations. While there is no perfect optimization algorithm, the hybridization of such metaheuristic optimization algorithms has produced positive results by enhancing their capabilities and reducing their weaknesses. This paper presents a novel hybridization of Particle Swarm Optimization and the Fireworks Algorithm for solving nonlinear equation systems. The experimental results obtained indicate that the proposed hybrid algorithm outperforms both Particle Swarm Optimization and the Fireworks Algorithm, as well as a previously developed hybridization of these algorithms.

Keywords: Computational intelligence · Particle swarm optimization · Fireworks Algorithm · Hybrid algorithms · Nonlinear equation systems

1 Introduction

Nonlinear equation systems are recognized as challenging problems to solve using traditional iterative numerical methods, whose effectiveness largely depends on the characteristics of the problem considered and the quality of the initial approximations taken, with non-convergence occurring relatively frequently.

In contrast, the use of population-based metaheuristic algorithms, such as Particle Swarm Optimization (PSO) and the Fireworks Algorithm (FWA), for solving this important class of problems, which have an extensive importance in fields such as chemistry, physics, engineering, and economics, has the advantage of being problem-independent, derivative-free, and not dependent on good initial estimates. However, there is no guarantee of convergence with these population-based stochastic algorithms, and each of them has different advantages and drawbacks, which motivates their hybridization to capitalize on their respective strengths and mitigate their weaknesses.

With this in mind, this article proposes a novel hybrid metaheuristic algorithm for solving systems of nonlinear equations that is the result of the combination of Particle Swarm Optimization and the Fireworks Algorithm.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14112, pp. 55–65, 2023.
https://doi.org/10.1007/978-3-031-37129-5_5

Population-based approaches, such as the hybrid algorithm proposed in this paper, can solve a system of n nonlinear equations by converting it into an n-dimensional nonlinear optimization problem by minimizing the sum of squares of the residuals, $F(\mathbf{x}) = \sum[f_i(x_1, \ldots, x_n)]^2$, or the sum of absolute values of the residuals, $F(\mathbf{x}) = \sum |f_i(x_1, \ldots, x_n)|$. This optimization-based approach for solving nonlinear equation systems was adopted, and the performance of the proposed hybrid algorithm is here evaluated.

The structure of this paper is as follows. The next section describes the PSO and FWA algorithms, which served as basis for the proposed algorithm. Section 3 describes briefly some simple ways to combine PSO and FWA, as well as a previous hybridization of these two algorithms. Section 4 presents the proposed algorithm, and the experimental setup and test problems used in this study are described in Sect. 5. The main results obtained are presented and discussed in Sect. 6. Finally, the conclusion is given in Sect. 7.

2 Background

2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO), proposed by Eberhart and Kennedy [3], is a population-based stochastic algorithm, based on swarm intelligence, that draws inspiration from the cooperative behavior of groups of animals when searching for food. In PSO, each particle represents a candidate solution to the problem, and "remembers" the best solution it found, as well as the one by its group. At each iteration, the particles update their velocity and position as follows:

$$\mathbf{v}_i^{t+1} = w \cdot \mathbf{v}_i^t + {r_1}_i^t \cdot c_1 \cdot (\mathbf{pbest}_i^t - \mathbf{x}_i^t) + {r_2}_i^t \cdot c_2 \cdot (\mathbf{gbest}^t - \mathbf{x}_i^t), \tag{1}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}, \tag{2}$$

where $\mathbf{v}$ is the velocity of the particle, $\mathbf{x}$ is its position, w is the inertia weight introduced by Shi and Eberhart [12], r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$, and c_1 and c_2 are the cognitive and social factors, respectively. These affect how much the particle is drawn to their own previous experience, compared to the one found by their group, with **pbest** and **gbest** representing the particle's best position and the group's best position, respectively.

This sharing of information is at the core of the flexibility of PSO. It allows not only changes to the algorithm itself, but also different topologies for the group of particles that communicate between themselves. The initial topology is the gbest topology, where all particles communicate with each other. Premature convergence was observed as a result of this [8].

The pseudocode of the standard PSO is given in Algorithm 1.

Other topologies that reduce the degree of communication have been proposed. One of these topologies is the von Neumann topology [6], where every particle is connected with a small fixed number of particles, typically four. This topology was found to improve the performance of the algorithm by allowing a more efficient search space exploration [8].

Algorithm 1. Pseudocode for the standard PSO algorithm

```
Initialize a swarm of N particles randomly;
while t < MaxIter and not terminate do
    for each particle position x_i in the swarm do
        Evaluate f(x_i)
        if f(x_i) < f(pbest) then
            Update the particle pbest
            if f(x_i) < f(gbest) then
                Update the particle gbest
            end if
        end if
    end for
    for i from 1 to N do
        Calculate v_i^{t+1} using Eq. 1;
        Calculate x_i^{t+1} using Eq. 2;
    end for
end while
Return the best particle found;
```

2.2 Fireworks Optimization

The Fireworks Algorithm (FWA) [13] is a population-based swarm intelligence algorithm inspired by the fireworks explosion, as each potential solution "explodes", creating a new sub-swarm. The amplitude of such explosion is dependent on the performance of the fireworks, where less-optimal solutions spread further apart, allowing for greater global exploration, whereas better performing fireworks spread much less, allowing for better local exploitation.

The amplitude of explosion for each firework can be calculated as follows:

$$A_i = \hat{A} \cdot \frac{f(\mathbf{x}_i) - y_{min} + \xi}{\sum_{i=1}^{N} (f(\mathbf{x}_i) - y_{min}) + \xi}, \tag{3}$$

where $\hat{A}$ is the maximum explosion amplitude, $f(\mathbf{x}_i)$ is the fitness value of the firework $\mathbf{x}_i$, $y_{min} = \min f(\mathbf{x}_i)\,(i = 1, \ldots, N)$ is the fitness value of the best firework, and ξ is an arbitrarily small number used to avoid division by 0.

The number of sparks produced by the exploding firework is given by:

$$S_i = m \cdot \frac{y_{max} - f(\mathbf{x}_i) + \xi}{\sum_{i=1}^{N} (y_{max} - f(\mathbf{x}_i)) + \xi}, \tag{4}$$

where m is a parameter that control the number of sparks, and y_{max} is as in (3).

Since the number of sparks must be an integer, there is an additional step that limits the number of sparks that can be generated as follows:

$$S_i = \begin{cases} round(a \cdot m) & \text{if } S_i < a \cdot m \\ round(b \cdot m) & \text{if } S_i > b \cdot m \quad (a < b < 1) \\ round(S_i) & \text{otherwise.} \end{cases} \tag{5}$$

Algorithm 2. Pseudocode for the FWA algorithm

```
Initialize a swarm of N fireworks randomly;
Evaluate fitness f(x) of each firework;
while t < MaxIter and not terminate do
    for each firework in the swarm do
        Calculate A_p according to Eq. 3;
        Calculate S_p according to Eq. 4 and Eq. 5;
        Generate and evaluate new sparks;
        for each spark do
            Evaluate the fitness of the spark;
            Select the best firework/sparks to be "Gaussian fireworks";
            Generate a few "Gaussian sparks" around the "Gaussian fireworks"
    based on a Gaussian distribution;
            Evaluate the fitness for each "Gaussian spark";
        end for
    end for
    Select the best points from the current fireworks and sparks to form the next
    generation of fireworks;
end while
Return the best firework found;
```

Algorithm 2 provides a pseudocode for the Fireworks Algorithm.

FWA has been proposed as a better alternative to PSO, and boasts a faster convergence. This is an advantage on some problems, but it can also lead to premature convergence, which was one of the reasons that motivated modifications to the PSO topology.

3 Related Work

PSO and FWA are population-based optimization approaches that can be combined to leverage their respective strengths. PSO is known for its simplicity and efficiency, while FWA is known for its ability to prevent local optima.

A simple way to combine these algorithms is to use FWA to initialize the particles in PSO, providing the swarm with a good starting point for exploration. Similarly, PSO can be used to influence the FWA explosion operation, directing the sparks to promising areas of the search space.

In the newly Dynamic Fireworks Algorithm with Particle Swarm Optimization (DFWPSO) [16], PSO was utilized in a new mechanism for update fireworks, aiming to accelerate the convergence of the FWA algorithm and reduce the computing time, thus improving the overall performance of the fireworks algorithm.

By appropriately combining the strong exploitation abilities of PSO with the strong exploration capabilities of FWA, it is possible to build more sophisticated hybrid algorithms that achieve a better balance between the exploration of the entire search space and exploitation, i.e., refinement of the best-known solutions.

In the PS–FW algorithm [2], modified FWA operators are incorporated into the PSO solution procedure. The concept behind this hybrid algorithm is that at

the start of each iteration, the velocity and position of each particle are updated in the same way as in the PSO algorithm, and then the so-called abandonment and supplement mechanism is applied aiming to balance the exploration and exploitation ability of the PS–FW algorithm. These two operations follow the logic of FWA. A number of particles with the worst fit are discarded as a result of the abandonment operation, while a number of particles with better fitness fit are retained for the subsequent iteration in view of the supplement operation.

These best particles are used to implement a modified explosion operator, a new mutation operator, and a fitness-based selection operator, of which the first two were developed to accelerate global convergence and prevent premature convergence to local optima. The new particles obtained by the FWA operators are added to the original swarm in order to balance the number of particles and generate a new particle swarm for the next algorithm iteration.

4 Proposed PSO–FWA Algorithm

Although the PS–FW algorithm [2] combines PSO and FWA and uses the local exploitation of PSO and the global exploration of FWA to achieve better results, the new PSO and FWA hybrid proposed in this paper, PSO-FWA, does not follow the same principle.

In fact, the new hybrid approach here presented does in a way the opposite, as PSO is here used for global exploration and FWA for local exploitation. In PSO–FWA, the von Neumann topology [6] is assumed to be used, but every few iterations, the *bp* best particles multiply using the amplitude and number of new particles (i.e., sparks) randomly generated within the amplitude A_i in the new swarm from FWA.

The PSO–FWA algorithm pseudocode is shown in Algorithm 3.

Whenever creating new particles dynamically in PSO, these new particles need belong to a group or sub-swarm, which can create problems in recalculating a new group for every particle. To avoid this, as well as to utilize the exploitation capabilities of FWA, since only the best particles get to reproduce in this way, there is not a large concern with premature convergence, so these new groups form a gbest topology with each other, as well as with their "parent" particle.

For a newly created particle to be able to communicate, it must know which particles belong to its group and make every existing particle aware of its existence. To maintain the von Neumann topology, this would mean recalculating the configuration of the entire swarm. Every time a particle multiplies, each new particle is connected to its parent and the other particles in the same sub-swarm via a gbest topology, in order to avoid this computational effort. As only a small number of particles multiply, and those selected to do so are the ones exploring the most promising regions, all new particles originate from such regions.

The use of the gbest topology for the new particles does not affect the general capacity of the swarm for global exploration, since the new particle only share information directly with its parent. This iterative creation of new sub-swarms connected to a single particle has an impact on the topology of the swarm.

Algorithm 3. Pseudocode for the proposed PSO–FWA algorithm

```
Initialize N particles with von Neumann topology;
Evaluate the fitness value f(x) of each particle;
while t < maxIter and not terminate do
    if t % k = 0 then
        Sort particles by fitness;
        y_min ← particles[1];
        y_max ← particles[N/2];
        best_particles ← particles[1 : bp];
        for p in bp do
            Calculate A_p according to Eq. 3;
            Calculate S_p according to Eq. 4 and Eq. 5;
            Generate new sub-swarm according to Algorithm 4;
        end for
    end if
    Calculate v_n^{t+1} for each particle according to Eq. 1;
    Calculate x_n^{t+1} for each particle according to Eq. 2;
    Evaluate fitness of each particle;
end while
Return the best particle found;
```

In the case of a problem with numerous local minima, where the particles are spread apart, the new sub-swarms will also be separated, allowing for a faster exploitation of the promising regions.

Algorithm 4. Pseudocode for the sub-swarm algorithm

```
for s in S_p do
    new_particles[s] ← p · Gaussian() · A_p;
    Connect all new_particles in same group;
end for
Return new_particles;
```

This does not impede the global exploration capabilities of the algorithm, as the new particles only indirectly share information to the rest of the particles while locally searching promising regions. This successive generation of new particles connected to a single parent particle slowly morphs the overall topology of the swarm. If the best particles are spread apart, indicating an objective function with many local minima, the particles that multiply will quickly explore these promising regions and further multiplications will abandon the groups that got stuck in local minima. By opposition, if the particles that multiply are close to each other, this will result in an increase in the number of particles in the promising regions, allowing for a fast convergence.

5 Experimental Setup

5.1 Experimental Setting

To allow for a fair comparison, the standard PSO, FWA, PS–FW, and PSO–FWA algorithms were implemented and compared in the same experimental setting. All algorithms were run 51 times each, as suggested in [7], using the same control parameters. The maximum number of iterations allowed for each algorithm was set to 1,000, the inertia weight w was set to 0.7, and the acceleration coefficients c_1 and c_2 were both set to 1.8. In all experiments, a tolerance of 1e−12 was adopted.

Regarding algorithm-specific parameters, the number of best particles *bp* was set to 5 and a new sub-swarm was generated every 50 iterations.

The von Neumann topology was used in both the standard PSO and PSO–FWA, whereas the gbest topology was utilized for PS–FW, as it was used in the original algorithm [2].

The values used for the specific parameters in the FWA and PS–FW algorithms were the same as in [2,13], respectively.

The comparison also included the Enhanced Jaya (EJAYA) algorithm [15], a variant of the Jaya metaphor-less algorithm [10] that has recently been demonstrated to be quite effective in solving systems of nonlinear equations [11].

The implementation of the algorithms was done in Julia [1] using double-precision floating-point arithmetic. All experiments were run on a portable computer with an Intel® Core™ i7-4720HQ processor at 2.60 GHz and 8 GB RAM.

5.2 Test Problems

The following systems of nonlinear equations, which are difficult to solve using traditional iterative methods, were chosen as test problems for the comparison performed in this study:

Problem 1. ([4], Problem D1 – Modified Rosenbrock), $n = 12$.

$$
\begin{aligned}
f_{2i-1}(\mathbf{x}) &= \frac{1}{1+\exp(-x_{2i-1})} - 0.73 \\
f_{2i}(\mathbf{x}) &= 10(x_{2i} - x_{2i-1}^2), \quad i = 1, \ldots, \frac{n}{2} \\
D &= ([-10, 10], \ldots, [-10, 10])^T
\end{aligned}
$$

Problem 2. ([4], Problem D3 – Powell badly scaled), $n = 12$.

$$
\begin{aligned}
f_{2i-1}(\mathbf{x}) &= 10^4 x_{2i-1} x_{2i} - 1 \\
f_{2i}(\mathbf{x}) &= \exp(-x_{2i-1}) + \exp(-x_{2i}) - 1.0001, \quad i = 1, \ldots, \frac{n}{2} \\
D &= ([0, 100], \ldots, [0, 100])^T
\end{aligned}
$$

Problem 3. ([4], Problem D6 – Shifted and augmented trigonometric function with an Euclidean sphere), $n = 12$.

$$f_i(\mathbf{x}) = n - 1 - \sum_{j=1}^{n-1} \cos(x_j - 1) + i(1 - \cos(x_i - 1)) - \sin(x_i - 1), \quad i = 1, \ldots, n-1$$
$$f_n(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 - 10000$$
$$D = ([-200, 200], \ldots, [-200, 200])^T$$

Problem 4. ([14], Economics modeling application), $n = 12$.

$$f_i(\mathbf{x}) = \left(x_i + \sum_{k=1}^{n-i-1} x_k x_{i+k} \right) x_n - c_i, \quad i = 1, \ldots, n-1$$
$$f_n(\mathbf{x}) = \sum_{j=1}^{n-1} x_j + 1$$

where the constants c_i can be chosen arbitrarily; here $c_i = 0$, $i = 1, \ldots, n-1$
$D = ([-100, 100], \ldots, [-100, 100])^T$

Problem 5. ([5], Example 1 – The Bratu problem), $n = 12$.

$$f_1(\mathbf{x}) = -2x_1 + x_2 + \alpha h^2 \exp(x_1)$$
$$f_n(\mathbf{x}) = x_{n-1} - 2x_n + \alpha h^2 \exp(x_n)$$
$$f_i(\mathbf{x}) = x_{i-1} - 2x_i + x_{i+1} + \alpha h^2 \exp(x_i), \ i = 2, \ldots, n-1,$$

where $\alpha \geq 0$ is a parameter, assuming here $\alpha = 3.5$, and $h = \dfrac{1}{n+1}$.
$D = ([-100, 100], \ldots, [-100, 100])^T$

Problem 6. ([5], Example 2 – The beam problem), $n = 12$.

$$f_1(\mathbf{x}) = -2x_1 + x_2 + \alpha h^2 \sin(x_1)$$
$$f_n(\mathbf{x}) = x_{n-1} - 2x_n + \alpha h^2 \sin(x_n)$$
$$f_i(\mathbf{x}) = x_{i-1} - 2x_i + x_{i+1} + \alpha h^2 \exp(x_i), \ i = 2, \ldots, n-1,$$

where $h = \dfrac{1}{n+1}$ and $\alpha \geq 0$ is a parameter; here $\alpha = 11$.
$D = ([-100, 100], \ldots, [-100, 100])^T$

Problem 7. ([9], 21 – Extended Rosenbrock function), $n = 12$.

$$f_{2i-1}(\mathbf{x}) = 10(x_{2i} - x_{2i-1}^2)$$
$$f_{2i}(\mathbf{x}) = 1 - x_{2i-1}, \quad i = 1, \ldots, \frac{n}{2}$$
$$D = ([-100, 100], \ldots, [-100, 100])^T$$

Problem 8. ([9], 26 – Trigonometric function), $n = 12$.

$$f_i(\mathbf{x}) = n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i, \quad i = 1, \ldots, n$$
$$D = ([-100, 100], \ldots, [-100, 100])^T$$

Problem 9. ([9], 27 – Brown almost-linear function), $n = 12$.

$$f_i(\mathbf{x}) = x_i + \sum_{j=1}^{n} x_j - (n+1), \quad i = 1, \ldots, n-1$$
$$f_n(\mathbf{x}) = \left(\prod_{j=1}^{n} x_j \right) - 1$$
$$D = ([-10, 10], \ldots, [-10, 10])^T$$

Problem 10. ([9], 28 – Discrete boundary value function), $n = 12$.

$$f_1(\mathbf{x}) = 2x_1 - x_2 + h^2(x_1 + h + 1)^3/2$$
$$f_n(\mathbf{x}) = 2x_n - x_{n-1} + h^2(x_n + nh + 1)^3/2$$
$$f_i(\mathbf{x}) = 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2, \; i = 2, \ldots, n-1,$$

where $h = \frac{1}{n+1}$ and $t_i = ih$.
$D = ([0, 5], \ldots, [0, 5])^T$

6 Results and Discussion

The average fitness value for each pair algorithm/problem was calculated and the results thus obtained are shown in Table 1. The best (i.e., minimum) fitness values found for each different algorithm and problem are shown in Table 2. The best result for each problem is bolded, while the second-best value is underlined.

Table 1. Average performance of each algorithm

Problem	PSO	FWA	PSO–FWA	PS–FW	EJAYA
01	0.093696602	8.231227429	**0.023829129**	2.688307333	0.786323905
02	**0.000611303**	10528.36554	0.053180425	4.929956145	5.845716462
03	**0.673563009**	17637.37896	0.813282593	7.37086e+15	24.94501395
04	7.68886e−13	121.161161	**3.50538e−13**	7.08888e+24	6.919e−13
05	2.472214572	405.3670455	**0.017626657**	0.180648169	18.40166643
06	**0.629464887**	449.2970073	0.708199283	0.554307947	16.3762457
07	**0.740239982**	1166.651425	1.946738341	4.30706e+11	11.87298727
08	**0.003830296**	0.032297347	0.03146428	0.025972228	4.244178222
09	3.08272e−05	27.63435204	**4.73538e−06**	4.77479e+66	0.019622826
10	0.001165984	0.156890307	**6.90023e−05**	7.32552e+14	0.237439165

Table 2. Best performance of each algorithm

Problem	PSO	FWA	PSO–FWA	PS–FW	EJAYA
01	**6.58917e−07**	0.214234267	2.67113E-05	0.616052907	0.005183943
02	0.0006113	837.2544034	**0.000413175**	2.278576919	5.000485975
03	0.00467257	3447.14932	**3.14072e−09**	3.68477048	11.79736641
04	1.64973e−13	1.93052557	2.79099e−13	**3.64338e−14**	1.2451e−13
05	2.469796642	45.3698909	**7.61427e−05**	0.061573931	1.01676417
06	0.517634554	80.45977863	**0.514741134**	0.518018226	1.212673401
07	4.71128E-07	60.61553916	**4.2838e−08**	3.047900675	2.048738198
08	0.003270911	0.021813477	**9.59711e−13**	6.0212E-05	1.282223168
09	9.7311e−13	1.41803206	**9.06386e−13**	0.29030721	3.5519e−06
10	**9.29977e−13**	0.111856778	9.8634e−13	0.045728666	0.237439165

From the results obtained, a few conclusions can be drawn. PSO–FWA achieved the best average result in half the problems, tied with PSO. While this may indicate a failure of the other algorithms, it should be noted that what these two algorithms have in common is the von Neumann topology, used to allow for a better exploration.

Due to their rapid convergence, the remaining algorithms do not perform well on challenging problems, such as nonlinear equation systems, for which exploration plays a crucial role.

Even though PSO on average performed similarly to the PSO–FWA algorithm, this approach has shown to have better convergence, as it achieved either the best result or a result quite close to the best in almost all problems.

7 Conclusion

PSO–FWA, a novel optimization algorithm based on a hybridization of PSO and FWA, was proposed to solve complex nonlinear equation systems. The performance of the proposed PSO–FWA hybrid algorithm was compared to PS–FW, another hybrid algorithm based on PSO and FWA, as well as to PSO and FWA.

The Enhanced Jaya (EJAYA) algorithm [15], which was recently shown quite effective at solving systems of nonlinear equations [11], was also included in the computational comparison carried out.

While the previous algorithms considered for comparison appear to perform better in simple optimization problems with few local minima, PSO–FWA was able to achieve better results in more difficult problems, as evidenced by the results obtained with the set of complex nonlinear equation systems chosen for this study.

References

1. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671

2. Chen, S., Liu, Y., Wei, L., Guan, B.: PS-FW: a hybrid algorithm based on particle swarm and fireworks for global optimization. Comput. Intell. Neurosci. **2018**, 6094685 (2018). https://doi.org/10.1155/2018/6094685
3. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43. IEEE (1995). https://doi.org/10.1109/MHS.1995.494215
4. Friedlander, A., Gomes-Ruggiero, M., Kozakevich, D., Martínez, J., Santos, S.: Solving nonlinear systems of equations by means of quasi-Newton methods with a nonmonotone strategy. Optim. Methods Softw. **8**(1), 25–51 (1997). https://doi.org/10.1080/10556789708805664
5. Kelley, C., Qi, L., Tong, X., Yin, H.: Finding a stable solution of a system of nonlinear equations. J. Ind. Manag. Optim. **7**(2), 497–521 (2011). https://doi.org/10.3934/jimo.2011.7.497
6. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC 2002 (Cat. No.02TH8600), vol. 2, pp. 1671–1676 (2002). https://doi.org/10.1109/CEC.2002.1004493
7. Liang, J., Qu, B., Suganthan, P., Hernández-Díaz, A.: Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China (2013)
8. Liu, Q., Wei, W., Yuan, H., Zhan, Z.H., Li, Y.: Topology selection for particle swarm optimization. Inf. Sci. **363**, 154–173 (2016). https://doi.org/10.1016/j.ins.2016.04.050
9. Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981). https://doi.org/10.1145/355934.355936
10. Rao, R.: Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. **7**, 19–34 (2016). https://doi.org/10.5267/j.ijiec.2015.8.004
11. Ribeiro, S., Silva, B., Lopes, L.G.: Solving systems of nonlinear equations using Jaya and Jaya-based algorithms: a computational comparison. In: Yadav, A., Nanda, S.J., Lim, M.H. (eds.) Proceedings of the International Conference on Paradigms of Communication, Computing and Data Analytics: PCCDA 2023. Springer, Singapore (to appear, 2023)
12. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), 1998, pp. 69–73. IEEE (1998). https://doi.org/10.1109/ICEC.1998.699146
13. Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: International Conference on Swarm Intelligence, pp. 355–364 (2010). https://doi.org/10.1007/978-3-642-13495-1_44
14. van Hentenryck, P., McAllester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. **34**(2), 797–827 (1997). https://doi.org/10.1137/S0036142995281504
15. Zhang, Y., Chi, A., Mirjalili, S.: Enhanced Jaya algorithm: a simple but efficient optimization method for constrained engineering design problems. Knowl. Based Syst. **233**, 107555 (2021). https://doi.org/10.1016/j.knosys.2021.107555
16. Zhu, F., Chen, D., Zou, F.: A novel hybrid dynamic fireworks algorithm with particle swarm optimization. Soft. Comput. **25**(3), 2371–2398 (2020). https://doi.org/10.1007/s00500-020-05308-6