Earthline Journal of Mathematical Sciences
ISSN (Online): 2581-8147
Volume 5, Number 2, 2021, Pages 267-275
https://doi.org/10.34198/ejms.5221.267275

Refined Heuristic Swarm Intelligence Algorithm

I. I. Aina[1,*] and C. N. Ejieji[2]

[1] Department of Mathematics, University of Ilorin, Ilorin, Nigeria
e-mail: ibkisaac@gmail.com

[2] Department of Mathematics, University of Ilorin, Ilorin, Nigeria
e-mail: ejieji.cn@unilorin.edu.ng

Abstract

In this paper, a new metaheuristic algorithm named refined heuristic intelligence swarm (RHIS) algorithm is developed from an existing particle swarm optimization (PSO) algorithm by introducing a disturbing term to the velocity of PSO and modifying the inertia weight, in which the comparison between the two algorithms is also addressed.

1 Introduction

For the last decade, many researchers in this field have changed direction, leaving aside traditional optimization techniques based on linear and nonlinear programming and embarked in the implementation of the evolutionary algorithm: genetic algorithm; ant colony optimization; simulated annealing; and harmony search among others. Particle swarm optimization (PSO) algorithm was first introduced by Kennedy and Eberhart in 1995 by observing the behavior of animals, e.g., bird flocking and fish schooling. Their movements and communication mechanisms were thoroughly studied [1, 2, 4].

In comparison with several other population based stochastic optimization methods such as genetic algorithm (GA) and evolutionary programming (EP), PSO performs better in solving various optimization problems with fast and stable convergence rate [3, 6].

Received: August 3, 2020; Accepted: September 2, 2020
2010 Mathematics Subject Classification: 90C59.
Keywords and phrases: particle swarm optimization, refined heuristic intelligence swarm, metaheuristic algorithm.
*Corresponding author

In this research work a new metaheuristic method is designed which performs better than the existing one.

2 Particle Swarm Optimization (PSO) Algorithm

The pseudo code for particle swarm algorithm (PSO) is as follows:

Step 1: Initialize the population size N_p, the dimension of the space
c_1, c_2, It_{max}, w_{max}, w_{min}

Step 2: Set $P_{best,i} = x_i, \qquad (i = 1, ..., N_p)$
calculate the fitness of x_i
Find G_{best}

Step 3: If $iter < It_{max}$

Step 4: Calculate the inertia weight
$w = [w_{max} - \dfrac{w_{max} - w_{min}}{It_{max}}(iter)]$

Step 5: Calculate the velocity of each particle by
$v^{t+1} = wv_i^t + c_1 r_1(P_{best,1} - x_i^t) + c_2 r_2(G_{best} - x_i^t)$

Step 6: Calculate the position of each particle by
$x_i^{t+1} = x_i^t + v_i^{t+1}$

Step 7: Calculate the value of the objective function for each particle
$f(x^{t+1})$

Step 8: Find $P_{best,i}^{t+1}$ and G_{best}
if $f_i^{t+1} > P_{best,i}^t$ then $P_{best}^{t+1} = P_{best,i}^t$
else
$P_{best,i}^{t+1} = x_i^{t+1}$

Step 9: Go to Step 3.

Table 2.1 shows the parameters used in the above PSO algorithm.

Table 2.1: Parameters used of PSO algorithm.

Parameter	Definition
c_1, c_2	PSO Constant, mostly given as 2
It_{max}	The maximum number of iteration
$iter$	The number of current iteration
w	Inertia weight
w_{max}	Maximum inertia weight
w_{min}	Minimum inertia weight
x_i	Position of particle i
N_p	Total number of particle
x_i^{t+1}	The position of particle i at $(t+1)^{th}$ iteration
x_i^t	The position of particle i at $(t)^{th}$ iteration
v_i^{t+1}	The velocity of particle i at $(t+1)^{th}$ iteration
$P_{best,i}^{t+1}$	The personal best of particle i at $(t+1)^{th}$ iteration
$G_{best,i}$	The global best
r_1, r_2	Random variable, which is always between o and 1

Algorithm of Refined Heuristic Swarm Intelligence (RHSI)

Just like other global algorithm, PSO converges prematurely and being trapped into a local minimum. This leads to the modification of PSO algorithm called refined heuristic swarm intelligence (RHSI) algorithm which is the major concentration of this paper.

3 Refined Heuristic Swarm Intelligence (RHSI) Algorithm

RHSI algorithm was developed as a result of the premature convergence of PSO algorithm and being trapped by local minimum.

In this algorithm the natural log of the inertial weight (w) of PSO is being considered as the inertia weight of the algorithm, i.e.,

$$w = ln[w_{max} - \frac{w_{max} - w_m in}{It_{max}}(iter)]. \tag{3.1}$$

Also a disturbing term is introduced to the velocity of PSO which makes the method more efficient. This disturbing term $|b(r_3 - 0.5)|$ is introduced to the velocity, where b is a small number and r is a random number in the range $(0, 1)$. We take $b = 0.05$ in this paper. Therefore the velocity of each particle is being updated with

$$v^{t+1} = wv_i^t + c_1 r_1(P_{best,i}^t - x_i^t) + c_2 r_2(G_{best} - x_i^t) + |b(r_3 - 0.5)|. \tag{3.2}$$

The pseudo code for algorithm of refined heuristic swarm intelligence (RHSI) is as follows:

Step 1: Initialize the population size N_p, the dimension of the space
c_1, c_2, It_{max}, w_{max}, w_{min}

Step 2: Set $P_{best,i} = x_i, \qquad (i = 1, ..., N_p)$
calculate the fitness of x_i
Find G_{best}

Step 3: If $iter < It_{max}$

Step 4: Calculate the inertia weight
$w = ln([w_{max} - \frac{w_{max} - w_m in}{It_{max}}(iter)])$

Step 5: Calculate the velocity of each particle by

$$v^{t+1} = wv_i^t + c_1 r_1(P_{best,i}^t - x_i^t) + c_2 r_2(G_{best} - x_i^t) + |b(r_3 - 0.5)|$$

Step 6: Calculate the position of each particle by

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Step 7: Calculate the value of the objective function for each particle

$$f(x^{t+1})$$

Step 8: Find $P_{best,i}^{t+1}$ and G_{best}

if $f_i^{t+1} > P_{best,i}^t$ then $P_{best}^{t+1} = P_{best}^t, i$

else

$P_{best,i}^{t+1} = x_i^{t+1}$

Step 9: Go to step 3.

4 Experiment and Discussion

4.1 Computational Consideration

In this paper we will take the following values for the PSO and RHSI parameters: $w_{max} = 0.9$, $w_{min} = 0.4$, $It_{max} = 1000$, $b = 0.05$, number of particles=10.

The numerical results are presented in Tables 4.1 and it was obtained from the solution of some test problems are presented in this section using MATLAB R2010a (7.10.499) run on the PC Intel(R) samsung, a 32 bit Os Laptop windows 7 operating system.

The following standard problems from [5], were used to test the performance of the new algorithm (RHSI):

Problem 1 (Extended Rosenbrock Function) [5]

$$f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2, \quad x_0 = [-1.2, 1, ..., -1.2, 1], \quad c = 100. \tag{4.1}$$

Problem 2 (Extended Himmelblau Function) [5]

$$f(x) = \sum_{i=1}^{n/2}(x_{2i-1}^2 + x_{2i} - 11)^2 + (x_{2i-1} - x_{2i}^2 - 7)^2, \quad x_0 = [1, 1, ..., 1]. \tag{4.2}$$

Problem 3 (Extended White & Holst Function) [5]

$$f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1}^3)^2 + (1 - x_{2i-1})^2, \quad x_0 = [-1.2, 1, ..., -1.2, 1], \quad c = 100. \tag{4.3}$$

Problem 4 (Generalized Tridiagonal-1 Function) [5]

$$f(x) = \sum_{i=1}^{n-1}(x_i + x_{i+1} - 3)^2 + (x_i + x_{i+1} + 1)^4, \quad x_0 = [2, 2, ..., 2]. \tag{4.4}$$

Problem 5 (Extended Tridiagonal-1) [5]

$$f(x) = \sum_{i=1}^{n/2}(x_{2i-1} + x_{2i} - 3)^2 + (x_{2i-1} - x_{2i} + 1)^4, \quad x_0 = [2, 2, ..., 2]. \tag{4.5}$$

Problem 6 (Generalized White & Holst Function) [5]

$$f(x) = \sum_{i=1}^{n-1} c(x_{i+1} - x_i^3)^2 + (1 - x_i)^2, \quad x_0 = [-1.2, 1, ..., -1.2, 1], \quad c = 100. \tag{4.6}$$

Problem 7 (Generalized PSC1 Function) [5]

$$f(x) = \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2 + x_i x_{i+1})^2 + sin^2(x_i) + cos^2(x_i), \quad x_0 = [3, 0.1, ..., 3, 0.1]. \tag{4.7}$$

Problem 8 (Extended Cliff Function) [5]

$$f(x) = \sum_{i=1}^{n/2}(\frac{x_{2i} - 3}{100})^2 - (x_{2i-1} - x_{2i}) + exp(20(x_{2i-1} - x_{2i})), \quad x_0 = [0, -1, ..., 0, -1]. \tag{4.8}$$

Problem 9 (Extended Hiebert Function) [5]

$$f(x) = \sum_{i=1}^{n/2}(x_{2i-1} - 10)^2 + (x_{2i-1}x_{2i} - 50000)^2, \quad x_0 = [0, 0, ..., 0]. \tag{4.9}$$

Problem 10 (Extended Quadratic Penalty QP1 Function) [5]

$$f(x) = \sum_{i=1}^{n-1}(x_i^2 - 2)^2 + (\sum_{i=1}^{n} x_i^2 - 0.5)^2, \quad x_0 = [1, 1, ..., 1]. \tag{4.10}$$

Problem 11 (Extended Quadratic Penalty QP2 Function) [5]

$$f(x) = \sum_{i=1}^{n-1}(x_i^2 - sinx_i)^2 + (\sum_{i=1}^{n} x_i^2 - 100)^2, \quad x_0 = [1, 1, ..., 1]. \tag{4.11}$$

Problem 12 (Extended Exponential EP1 Function) [5]

$$f(x) = \sum_{i=1}^{n/2}(exp(x_{2i-1} - x_{2i}) - 5)^2 + (x_{2i-1} - x_{2i})^2(x_{2i-1} - x_{2i} - 11)^2,$$
$$x_0 = [1.5, 1.5, ..., 1.5]. \tag{4.12}$$

Problem 13 (Diagonal 8 Function) [5]

$$f(x) = \sum_{i=1}^{n} x_i exp(x_i) - 2x_i - x_i^2, \quad x_0 = [1, 1, ..., 1]. \tag{4.13}$$

Problem 14 (GENHUMPS Function) [5]

$$f(x) = \sum_{i=1}^{n-1} sin(2x_i)^2 sin(2x_{i+1})^2 + 0.05(x_i^2 + x_{i+1}^2), \quad x_0 = [-506.2, 506.2, ..., 506.2]. \tag{4.14}$$

4.2 Computational Results

The computational result is given in Table 4.1.

Table 4.1: Result obtained with RHSI and PSO.

		RHSI	PSO
Problem	DIM(n)	$f(x)$	$f(x)$
1	10	1.1398	7.9799
2	10	530.0000	554.0000
3	10	0.0000	0.0000
4	10	11.9504	7.5922
5	10	0.7128	100.4064
6	10	0.0045	0.6551
7	10	7.9907	8.9901
8	10	1.7063	1.0034
9	10	1249950000.0000	1249950410.0000
10	10	33.7192	33.6128
11	10	72000.2262	72910.3238
12	10	33.7192	33.6128
13	10	-4.0214	-4.8045
14	10	-2.531	-1.3745

4.3 Discussion on Numerical Results

From the result tabulated in Table 4.1, it can be seen that RHIS proposed in this paper performs better than PSO in terms of value of the objective function.

5 Conclusion

To improve the performance of PSO algorithm, a modified PSO algorithm (RHSI algorithm) based on disturbing term is introduced in this paper. Two improvement strategies are proposed to generate a new position for each particle, which the strategies are modifying the inertia weight and the velocity of the particle of which it performs better than PSO algorithm. Therefore RHSI algorithm is recommended for solving optimization problems.

References

[1] Chiabwoot Ratanavilisagul and Boontee Kruatrachue, A modified particle swarm optimization with mutation and reposition, *International Journal of Initiative Computing, Information and Control* 10 (2014), 2127-2142.

[2] Chunfeng Wang and Wenxin Song, A modified particle swarm optimization algorithm based on velocity updating mechanism, *Ain Shams Engineering Journal* 10 (2019), 847-866. https://doi.org/10.1016/j.asej.2019.02.006

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1979.

[4] J. Kennedy and R. Eberhart, Particle swarm optimization, *IEEE International Conference on Neural Networks* 4 (1999), 1942-1947.

[5] Neculai Andrei, An unconstrained optimization test functions collection, *Advanced Modeling and Optimization* 10 (2008), 147-161.

[6] Xueying Lv, Yitian Wang, Junyi Deng, Guanyu Zhang and Liu Zhang, Improved particle swarm optimization algorithm based on last-eliminated principle and enhanced information sharing, *Computational Intelligence and Neuroscience* 2018 (2018), Article ID 5025672. https://doi.org/10.1155/2018/5025672