Journal of Computer Science

Original Research Paper

Using the Cuckoo Search for Generating New Particles in Particle Swarm Optimization Algorithm

[1]Fariaa Abdalmajeed Hameed, [2]Harith Raad Hasan, [3]Ahmed Abdullah Ahmed and [4]Gulala Ali Hama Amin

[1]Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Iraq
[2]Sulaimani Polytechnic University, Kurdistan Technical Instiute Sulaimani, Iraq
[3]Faculty of Engineering and Science, Qaiwan International University (QIU) Raparin, Sulaymaniyah, Kurdistan Region- Iraq
[4]Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Iraq

Article history
Received: 27-01-2020
Revised: 11-03-2020
Accepted: 13-04-2020

Corresponding Author:
Fariaa Abdalmajeed Hameed
Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Iraq
Email: fariaa.hameed@spu.edu.iq

Abstract: This study is focused on as Cuckoo Search (CS), one of the current meta-heuristic optimization algorithm. The CS algorithm is useful in generating and searching for the most optimum particles of important meta-heuristic optimization algorithm, known as the Particle Swarm Optimization (PSO), to enhance its performance. This optimization is confirmed through a benchmark online optimization and actual problems. The PSO algorithm performance is also compared with differing algorithms representative of the area. The CS optimal solutions outperform alternative current solutions as CS has distinct search features. The study findings have implications for future studies and practice.

Keywords: Optimization, Cuckoo Search (CS), Particle Swarm Optimization

Introduction

In the current business landscape, there exist several optimization opportunities, with some of the top being the optimizing schedules and workflow for the mitigation of resources (cost and time) and maximization of output.

In this regard, simple exhaustive search constitute one of the pioneering optimizer types, with every potential combination presented to obtain the best one. This type of optimization is known for its accuracy as they best combination are eventually obtained but its efficiency is quite low because if there are more than a few thousand combinations, it takes considerable time to examine all of them. This is the reason behind the limiting of exhaustive search optimizers of the number of variables employed, or limiting the number of variable values-but the best alternative is by using Cuckoo Search (CS). Specifically, CS is deemed to an evolutionary optimization algorithm developed and presented by Yang and Deb (2009; Yang, 2011). CS primarily refers to the cuckoo bird, where the actual bird takes advantage of some other bird species by laying its eggs in their nests (Payne *et al*., 2005). In this case, every egg is a representation of a new solution-with the objective being the alternative use of new and potentially better solutions than those in existing ones. CS algorithm is prominent and popular owing to its simplicity.

Contrastingly, conventional optimization algorithms entail the use of traditional methods like the dynamic programming, branch-and-bound and gradient-based techniques whereas their modern counterparts involve meta-heuristics searching. Meta-heuristics algorithms include simulated annealing, evolutionary computation EC, colony optimization, among others.

In the past few years, evidence shows that meta-heuristic algorithm has been understudied although its combination with other optimization methods could create a robust system that could handle actual large scale problems. This holds true for the meta-heuristic algorithm branch known as the Particle Swarm Optimization (PSO). PSO refers to a stochastic search procedure that has its basis on social behavior observations (e.g., flocking birds and schools of fish). This algorithm type evaluates search space in a simultaneous manner through the use of global information, which is why it has a higher likelihood to determine global solution to a specific issue.

Previous Studies on CS Application

Under this section the CS application findings in prior studies are presented and discussed. On the basis of the reviewed work, there are several categories using CS and they include the field of engineering (Vaisakh *et al*., 2009; Fan and Zahara, 2007; Omran *et al*., 2005; Price *et al*., 2005; Qin *et al*., 2009), pattern recognition (Kennedy *et al*.,

2001; Potter *et al*., 2010; Xin *et al*., 2011; Storn and Price, 1997), software testing and data generation (Guo *et al*., 2006; Storn and Price, 1997; Hao *et al*., 2007), networking (Shelokar *et al*., 2007; Pant *et al*., 2008), job scheduling (Fu *et al*., 2010; El Dor *et al*., 2012) as well as data fusion and wireless sensor networks (Xin *et al*., 2010; Akbari and Ziarati, 2008).

More specifically, a new diagnosis technique that used CS in engineering was adopted in (Yang and Deb, 2010) to resolve issues of engineering design optimization, with the inclusion of springs design and welded beam structures. The optimization objective in this case is to mitigate the spring weight and the overall cost of fabrication. The findings were benchmarked with evolutionary methods (e.g., GA and PSO) and were evidenced to be quite efficient and superior in almost the entire issues tested. In (Vazquez, 2011), the spiking neuron's accuracy and performance was measured using pattern recognition/classification fitted with CS Algorithm and a comparison was later conducted between CS and DE algorithms performance. Based on the outcome, spiking neuron model trained with CS outperformed in the adjustment of synaptic neuron weights.

Moving on to (Bacanin, 2012), the author focused on process parameters including, maximum cycle number of nests, runtime, number of parameters and probability and their role in optimization of process performance measures. The study employed CS in object-oriented software to address issues of unconstrained optimization-combinational and numeric optimization issues in JAVA programming language. Based on the outcome of the benchmark test, the proposed software had good performance, indicating the readiness of the system to be used to solve new problems.

Natarajan and Subramanian (2012), the author brought forward Enhance Cuckoo Search for Optimization of Bloom Filter in Spam Filtering. The study used an Enhanced Cuckoo Search (ECS) algorithm to mitigate the total membership invalidation cost of BFs, within which the optimal false positive rates are determined along with the number of elements within each of the bins. Based on the experimental outcome, for CS and ECS outperformed in numbers of bins ECS. In a related study, (Burnwal and Deb, 2012) brought forward Cuckoo Search for the purpose of scheduling optimization of flexible manufacturing system by decreasing the penalty cost and increasing the machine use time as used performance. The finding was compared to evolutionary techniques findings (e.g., GA) and it indicated the efficient and better performance of CS when solving tested problems.

Moreover, the Cuckoo Based Particle Approach (CBPA) was also used in (Dhivya *et al*., 2011) to realize energy efficiency in Wireless Sensor Network and multimodal objective functions. CS was used to cluster head selection and form clusters among the sensor nodes, after which the authors measured the decrease in energy of Wireless Sensor Networks (WSNs) and the increase in the lifetime through performance. The CBPA was then compared with the standard LEACH protocol and HEED protocol. The results of the simulation illustrated that CBPA generated comparable results owing to the process of optimal search in cluster formation and appropriation of paths in transmitting sensed data.

Along the same line of study, (Yildiz, 2012) made use of CS for the selection of optimal parameters in the context of milling operations, (Chifu *et al*., 2012) optimized the composition processes of semantic web service with the assistance of CS, while (Kumar and Chakarverty, 2011) realized optimal design for reliable integrated system. Lastly, Walton *et al*. (2011) modified CS to solve non-linear issues (e.g., mesh generation).

Overview of Cuckoo Search Algorithm

Cuckoo Search (CS) is described as a novel meta-heuristic algorithm created to issues relating to optimization, aligned with its namesake, which is the parasitic behavior of cuckoo species, Levy flight behavior and fruit flies. Specifically, in CS, the cuckoo's walking steps are determined by the flights, with each egg representing a new solution and the objective being to utilize optimum ad better solutions replacing the ineffective ones in the nest, with each nest having one egg. It is possible to extend the algorithm to more complicated cases, within which each nest has many eggs (i.e., many solutions). For instance, when producing new solutions $x^{(t+1)}$ for cuckoo I, a Levy flight is performed using the following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \beta \oplus L\hat{e}vy(\lambda)$$

In the above equation, $\alpha > 0$ represents the step size that is related to the examined problem scales, where in majority of cases $\alpha = 1$ can be used. The product $\oplus$ represents entry wise multiplications, whose product is similar to the PSO, with the difference being that the random walk through Levy flight works more efficiently in terms of exploration of search step as the length of the step is longer in long-term. In essence, the Levy flights provide a random walk represented as follows:

$$L\hat{e}vy(\lambda) \sim u = t^{-\lambda}, (1 < \lambda \leq 3)$$

The above equation is an infinite variance, having an infinite mean. Figure 1 demonstrates the pseudo code of CS.

```
begin
        Objective function f(x)
        Generate initial population of n host nest
        Evaluate fitness and rank eggs
        while (t > MaxGeneration) or stop criterion
            t = t +1
            Get a cuckoo randomly/generate new solution by lèvy flights
            Evaluate quality/fitness, F_i
            Choose a random nest j
            if (F_i > F_j)
               Replace j by the new solution
            end if
            Worst nest is abandoned with probability P_a and new nest is built
            Evaluate fitness and rank the solutions and find current best
        end while
            Post process results and visualization
end
```

Fig. 1: Cuckoo search pseudo code

Parameters:
n represents the number of hosts nests,
P_a represents the probability of discovering an alien egg,
MaxIter represents the maximum number of iterations,
Initialization,
Generate initial n host, $X_i^{(T)}$
Evaluate $f(X_i^{(T)})$
Iterations:
New Solution Generation

$$x_i^{(t+1)} = x_i^{(t)} + \beta \oplus L\hat{e}vy(\lambda)$$

Evaluate $f(X_i^{(T+1)})$:

Select a random nest x_J

If $f(X_i^{(T)}) < f(X_i^{(T+1)})$

Replace $(x^{(T)})$ with $(x^{(t+1)})$
Discard a fraction of p_a worse nests,
Develop new nests with Levy flights,
Identify and keep the best solutions

Overview of Particle Swarm Optimization (PSO)

The Congress on Evolutionary Coputation (Kennedy and Eberhart, 1995) presented a paper on PSO in 1995, triggering waves of publications in the past decade on the different applications success of PSO to resolve issues concerning optimization, as inspired by the flocking and foraging behavior of birds and fish (Brownlee, 2011). Such attributes are highly desirable, easily understandable and implemented. PSO is mainly used for its timely convergence, particularly when pitted against other optimization algorithms (e.g., simulated annealing and genetic algorithms) as illustrated in (Abraham *et al*., 2006).

The appeal of PSO stems from its simple conceptual framework and the birds flocking analogy facilitating conceptual visualization of the search process. In particular, a solution in PSO is displayed through a particle, with the population of solutions referred to as swarm of particles and each particle having two major properties, namely position and velocity. Added to this, each particle shifts to a new position with the help of velocity and once they are settled in a new position, the best position of the particle and swarm are made current as required. Each particle velocity is modified on the basis of its experiences.

In other words, the velocity (V_i) of each particle is updated with the help of the following equation:

$$v_i^{t+1} = wv_i^t + c_1 \times rand \times \left(pbest_i - x_i^t\right)$$
$$+c_2 \times rand \times \left(gbest - x_i^t\right)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

In the above equation:

v_i^t = Represents the particle i velocity, with iteration t
w = Represents the weighing function
C_1 = Represents individual coefficient
C_2 = Represents social coefficient
Rand = Represents random number (0-1)
X_i^t = Represents the current position of particle i, at t iteration
v_i^{t+1} = Represents the current velocity of i particle, at t+1 iteration
$pbest_i$ = Represents the *pbest* of agent i at t iteration
gbest = Represents the best solution from the alternative solutions.

```
For each particle
{
    Initialize particle
}
Do until maximum iterations or minimum error criteria
{
     For each particle
     {
          Calculate Data fitness value
          If the fitness value is better than pBest
          {
              Set pBest = current fitness value
          }
          If pBest is better than gBest
          {
              Set gBest = pBest
          }       }
          For each particle
     {
          Calculate particle Velocity
          Use gBest and Velocity to update particle Data
     }
```

Fig. 2: PSO pseudo code

Repetitive steps of the process are carried out until a criterion is met. In the first step, PSO is initialized, where the initial swarm of particles is produced and each particle initialized at random position as well as velocity, after which each particle is evaluated for their fitness. Each calculation of the fitness value is compared against the prior optimum one of the particle and the prior optimum value of the whole swarm. From this, the personal best and global best positions are updated accordingly.

In case a stopping criterion falls short of being met, the velocity and position are adjusted to create a new swarm and the personal and global best positions and old velocity used in the update of velocity. The two main PSO operations are velocity update and position update, with the former being based on three components, namely, the old velocity constituting inertia/momentum term, experience of individual particle constituting cognitive or self-learning term and experience of the whole swarm constituting group/social learning term. In each term, a weight constant is allocated and for the fundamental PSO algorithm, there are three required constants. Notably, PSO algorithm is not in need of sorting of fitness values of solutions in any step. The pseudo-code of PSO is depicted in Fig. 2.

Particle Swarm Optimization/CS: PSO/CS and Frameworks

The proposed algorithm primarily aims to improve the PSO algorithm performance to achieve optimum solutions in comparison to standard PSO and CS algorithms with lower execution periods compared to standard PSO. Both CS and PSO algorithms have a stochastic nature and as such, are invaluable in achieving global optimum compared to their gradient descent counterpart. However, it is simple for the former to drop into the local optima with unsatisfactory convergence accuracy.

```
For each particle
{
Starting CS to generate the particle position and velocity vectors
}
Do until maximum iterations or minimum error criteria
{
      For each particle
      {
           Calculate Data fitness value
           If the fitness value is better than pBest
           {
              Set pBest = current fitness value
           }
           If pBest is better than gBest
           {
               Set gBest = pBest
           }       }
           For each particle
      {
           Calculate particle Velocity
           Use gBest and Velocity to update particle Data
      }
```

Fig. 3: CS/PSO pseudo code

Table 1: Dimensions, ranges, of benchmark test functions used in the experiments.

Test function	Dimension (n)	Range
F1	3	$x1$[10,55], $x2$[1.1,2], $x3$[10,40]
F2	2	$x1$ [17.5,40], $x2$[300,600]
F3	4	[12, 60]
F4	3	$x1$[0.02,0.8], $x2$[10,40], $x3$[3000,20000]
F5	30	[-100, 100]
F6	30	[-5.12, 5.12]
F7	30	[-32.768, 32.768]
F8	30	[-30,30]

In order to leverage the advantage using the best of CS and PSO to enhance optimization performance, the new design preservers the PSO algorithms. It uses CS to produce the initial values of the particles and the velocity vectors rather than randomly generating them with high cost and significant period of time, or to face the risk of being trapped in the local optima-thus, the CS strategy initialization of the PSO particles and velocity is conducted to improve PSO performance. The usual PSO algorithm proceeds right after.

The pseudo code of CS/PSOE is demonstrated in Fig. 3, where it is notable that the particles positions are produced through CS algorithm after which they are updated normally using PSO foe each particle, evaluated on the basis of their fitness value. Lastly, the optimum solution is identified by the algorithm.

Experimental Results and Discussion

The CS/PSO performance in light of minimization and maximization benchmark functions are chosen, with the inclusion of 4 actual problems (section 6.1 details the benchmark).

Range and *n*, are feasible bound as presented along with function dimension in Table 1.

Benchmark Functions

1. Gas Transmission (F1):

$$Min\, f(x) = \begin{cases} 8.61*10^5 x_1^{1/2} x_2 x_3^{-2/3} \left(x_2^2 - 1\right)^{-1/2} + 3.69*10^4 x_3 \\ +7.72*10^8 x_1^{-1} x_2^{0.219} - 765.43*10^6 x_1 . \end{cases}$$

2. Optimal Capacity of Gas Production Facilities (F2):

$$Min\, f(x) = \begin{cases} 61.8 + 5.72x_1 + 0.2623\left[\left(40 - x_1\right)\ln\left(\frac{x_2}{200}\right)\right]^{-0.85} \\ +0.087\left(40 - x_1\right)\ln\left(\frac{x_2}{200}\right) + 700.23x_2^{-0.75}. \end{cases}$$

3. Design of Gear Train (F3):

$$Min\, f(x) = \left\{\frac{1}{6.931} - \frac{T_d T_b}{T_a T_f}\right\}^2 = \left\{\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}\right\}^2 .$$

4. Optimal Thermo hydraulic Performance of an Artificially Roughened Air Heater (F4):

$$Max\, L = 2.51\ln e^+ + 5.5 - 0.1R_M - G_H .$$

where, $R_M = 0.95x_2^{0.35}$, $G_H = 4.5(e^+)^{0.28}(0.7)^{0.57}$, $e^+ = x_1x_3(\bar{f}/2)^{1/2}$, $\bar{f} = (f_s + f_r)/2$, $f_s = 0.079x_3^{-0.25}$, $f_r = 2(0.95x_3^{0.53} + 2.5\ln(1/2x_1)^2 - 3.75)^{-2}$

5. SpShpere (n variables) (F5):

$$Sp_n(\vec{x}) = \sum_{i=1}^{n} \chi_i^2$$

6. Rastrigin's Function (F6):

$$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos\left(2\pi x_i\right)\right]$$

7. Ackley's Function (F7):

$$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(cx_i\right)\right) + a + \exp(1)$$

where, $a = 20$, $b = 0.2$, $c = 2*\pi$

8. Rn Rosenbrock (F8):

$$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2\right].$$

Results and Discussion

The algorithm brought forward in this study is compared with both CS and PSO standard algorithms, with the parameters of PSO established as number of particles and number of iterations during the run C1, C2 = 1.19, R1, R2 = 1. With regards to the CS parameters, they were established as population size and number of iteration and problem dimension during the run, with maximum population size of 100, Cr = 0.5 and F = 0.7. Each problem has a distinct average of the best value for 20 run times and maximum number of functions evaluations (Nb.evals) (refer to Tables 2 and 3).

On the basis of the experiments, it is notable that CS/PSO managed to achieve the optimum outcomes on majority of the problems posed, with the algorithm leading to significant developments compared to prior PSO and CS, with CS/PSO evidently outperforming all other algorithms tested on Figures 4, 5, 6 and 7.

Figure 8 illustrates the optimum solution for CS/PSO algorithm, with iteration 4000 and 500 and domain (n = 30), for SpShpere function. Figures 9 and Fig. 10 illustrate the minimum result for the proposed algorithm CS/PSO, with iteration 3000 and 4000 and domain (n = 30), for both the Rastrigin function and Ackley function. The best value was obtained with iteration 1500 for Rn Rosenbrock, with domain {n = 30). Thus, performance was good in the overall prior functions.

Table 2: Comparison between the proposed method and different algorithms based on real life problem benchmark functions

Function	Nb.evals	CS	PSO	CS/PSO
F1	24000	1.6929e+006	1.6894e+006	7.43233e+006
F2	16000	1.6987e+002	1.6988e+002	1.69844e+002
F3	32000	0.9922e-008	0.9623e-008	1.40108e-010
F4	24000	4.2053e-005	4.1986e-005	2.31987e-006

Table 3: Comparison of classical PSO and CS with proposed CS/PSO on standard test function benchmark

Function		PSO	CS	CS/PSO
F5	Best	2.131×10^{-12}	1.4364×10^{-12}	2.2962×10^{-24}
	Worst	4.2322×10^{-2}	8.7664×10^{-7}	1.1863×10^{-11}
	Std	6.6875×10^{-3}	1.3406×10^{-7}	1.9369×10^{-12}
F6	Best	32.8287	21.2728	13.9294
	Worst	127.3341	175.3698	65.6672
	Std	19.0037	35.8085	9.2816
F7	Best	7.3543×10^{-6}	8.4260×10^{-8}	7.4252×10^{-13}
	Worst	5.9074	2.3162	1.5017
	Std	1.2183	0.6521	0.3366
F8	Best	2.0718	18.3952	0.0248
	Worst	485.0228	577.7486	225.5721
	Std	76.5658	91.5654	41.6379

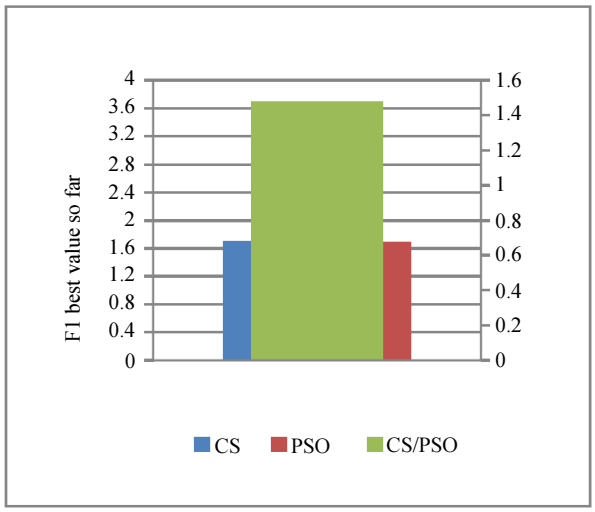

Fig. 4: Performance comparisons of CS, PSO, CS/PSO for function F1

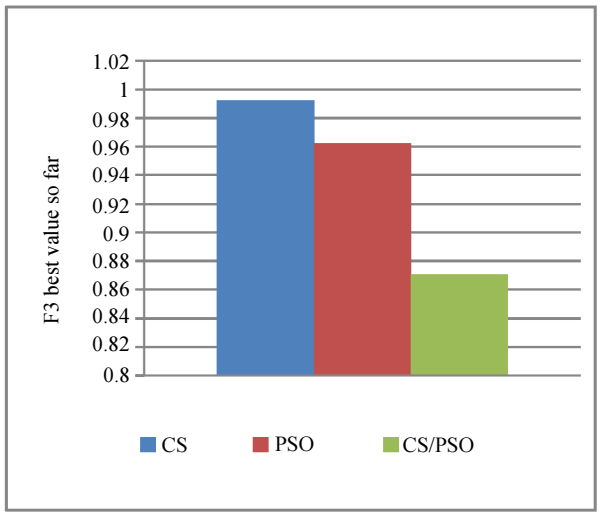

Fig. 6: Performance comparisons of CS, PSO, CS/PSO for function F3

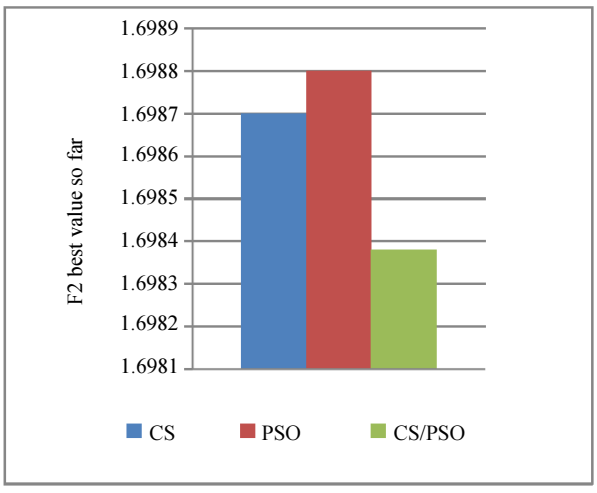

Fig. 5: Performance comparisons of CS, PSO, CS/PSO for function F2

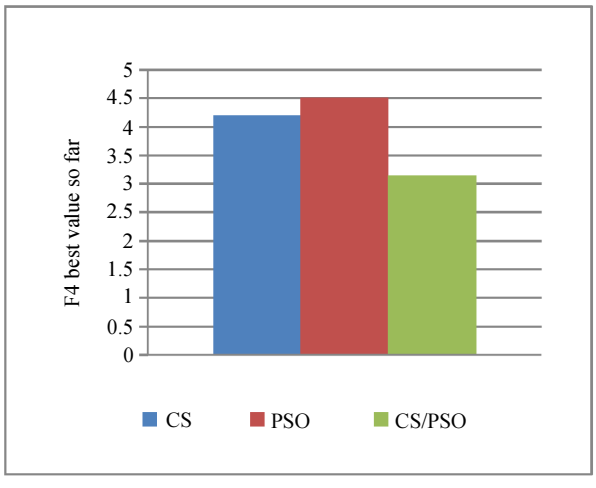

Fig. 7: Performance comparisons of CS, PSO, CS/PSO for function F4

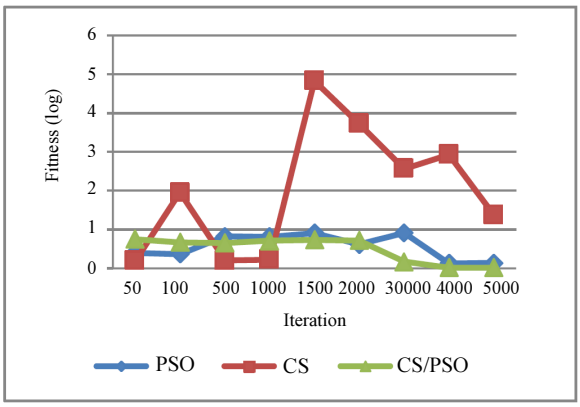

Fig. 8: The evolution curve of sphere function

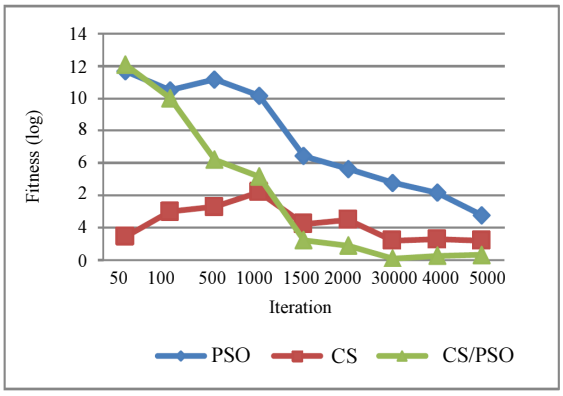

Fig. 9: The evolution curve of Rastrigin's function

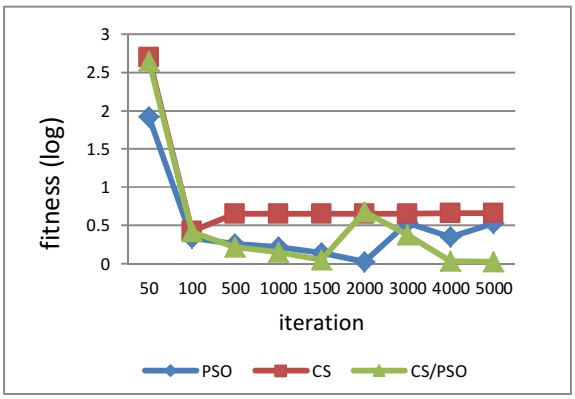

Fig. 10: The evolution curve of Ackley's function

Conclusion

The performance of PSO can be enhanced by achieving the optimum values and accordingly, we used the CS algorithm in this study to provide a balance between exploitation and exploration. The new method proposed, which is CS/PSO enhances the performance of particle swarm optimization through the incorporation of CS algorithm. The primary objective is to use the proposed algorithm to produce PSO particles, with the help of CS algorithm to improve the PSO searching abilities. The approach achieved outstanding results on 8 well-known benchmark functions, with the inclusion of 4 actual life problems. The results of the computations indicated that the proposed approach provided accurate outcomes in comparison to its evolutionary counterparts.

Acknowledgement

The authors would like to acknowledge the Sulaimani Polytechnic University, Qaiwan International University and Kurdistan Technical institute for their financial support to this research. Thanks also go to the reviewers of this paper for their constructive comments.

Author's Contributions

All authors equally contributed to this work.

Ethics

The Authors declare that there are no ethical issues associated with this work.

References

Abraham, A., H. Guo and H. Liu, 2006. Swarm Intelligence: Foundations, Perspectives and Applications. In: Swarm Intelligent Systems, Nedjah, N. and L. Mourelle (Eds.), Springer Berlin/Heidelberg, ISBN-13: 978-3-540-33869-7, pp: 3-25.

Akbari, R. and K. Ziarati, 2008. Combination of particles swarm optimization and stochastic local search for multimodal function optimization. Sci. China Inform. Sci., 53: 980-989. DOI: 10.1109/PACIIA.2008.84

Bacanin, N., 2012. Implementation and performance of an object-oriented software system for cuckoo search algorithm. Int. J. Math. Comput. Simulat., 6: 185-193.

Brownlee, J., 2011. Clever Algorithms: Jason Brownlee.

Burnwal, S. and S. Deb, 2012. Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. Int. J. Adv. Manufact. Technol.

Chifu, V., C. Pop, I. Salomie, D. Suia and A. Niculici, 2012. Optimizing the semantic web service composition process using cuckoo search. Intell. Distributed Comput.

Dhivya, M., M. Sundarambal and L.N. Anand, 2011. Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach. Int. J. Commun. Network Syst. Sci., 4: 249-255. DOI: 10.4236/ijcns.2011.44030

El Dor, A., M. Clerc and P. Siarry, 2012. Hybridization of differential evolutionand particle swarm optimization in a new algorithm: DEPSO-2S. Springer- Verlag Berlin Heidelberg, pp: 57-65.

Fan, S.K. and E. Zahara, 2007. A hybrid simplex search and particle swarm optimization for unconstrained optimization. Eur. J. Oper. Res. 181: 527-548. DOI: 10.1016/j.ejor.2006.06.034

Fu, W., M. Johnston and M. Zhang, 2010. Hybrid particle swarm optimization algorithms based on differential evolution and local search. Proceedings of the Australasian Joint Conference on Artificial Intelligence, (CAI' 10), Springer, Berlin, pp: 313-322. DOI: 10.1007/978-3-642-17432-2_32

Guo, Q.J., H.B. Yu and A.D. Xu, 2006. A hybrid PSO-GD based intelligent method for machine diagnosis. Digital Signal Process., 16: 402-418. DOI: 10.1016/j.dsp.2005.12.004

Hao, Z.F., G.H. Guo and H. Huang, 2007. A particle swarm optimization algorithm with differential evolution. Proceedings of the 6th International Conference on Machine Learning and Cybernetics, Aug. 19-22, IEEE Xplore Press, Hong Kong, China, pp: 1031-1035. DOI: 10.1109/ICMLC.2007.4370294

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proceedings of the International Conference on Neural Networks, (CNN' 95), pp: 1942-1948.

Kennedy, J., R.C. Eberhart and Y. Shi, 2001. Swarm Intelligence. 1st Edn., Morgan Kaufmann Publishers Inc.340 Pine Street, Sixth FloorSan FranciscoCAUnited States, ISBN-13: 978-1-55860-595-4.

Kumar, A. and S. Chakarverty, 2011. Design optimization for reliable embedded system using cuckoo search. Proceedings of the 3rd International Conference on Electronics Computer Technology, Apr. 8-10, IEEE Xplore Press, Kanyakumari, India, pp: 264-268. DOI: 10.1109/ICECTECH.2011.5941602

Natarajan, A. and P.K. Subramanian, 2012. An enhanced cuckoo search for optimization of bloom filter in spam filtering. Global J. Comput. Sci. Technol.

Omran, M.G.H., A.P. Engelbrecht and A. Salman, 2005. Differential evolution methods for unsupervised image classification. Proceedings of the Congress on Evolutionary Computation, Sept. 2-5, IEEE Xplore Press, Edinburgh, Scotland, UK. DOI: 10.1109/CEC.2005.1554795

Pant, M., R. Thangaraj, C. Grosan and A. Abraham, 2008. Hybrid differential evolution-particle swarm optimization algorithm for solving global optimization problems. Proceedings of the 3rd International Conference on Digital Information Management, Nov. 13-16, IEEE Xplore Press, London, UK, pp: 18-24. DOI: 10.1109/ICDIM.2008.4746766

Payne, R.B., M.D. Sorenson and K. Klitz, 2005. The Cuckoos. 1st Edn., Oxford University Press, Oxford, New York.

Potter, C., G.K. Venayagamoorthy and K. Kosbar, 2010. RNN based MIMO channel prediction. Signal Process., 90: 440-450. DOI: 10.1016/j.sigpro.2009.07.013

Price, K., R.M. Storn and J.A. Lampinen, 2005. Differential Evolution: A Practical Approach to Global Optimization. 1st Edn., Springer Science and Business Media, ISBN-10: 3540313060, pp: 539.

Qin, A.K., V.L. Huang and P.N. Suganthan, 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput., 13: 398-417. DOI: 10.1109/TEVC.2008.927706

Shelokar, P.S., P. Siarry, V.K. Jayaraman and B.D. Kulkarni, 2007. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Applied Math. Comput., 188: 129-142. DOI: 10.1016/j.amc.2006.09.098

Storn, R. and K. Price, 1997. Differential evolution-A simple and efficient heuristics for global optimization over continuous spaces. J. Global Optim., 11: 341-359. DOI: 10.1023/A:1008202821328

Vaisakh, K., P. Praveena and S.R.M. Rao, 2009. DEPSO and bacterial foraging optimization based dynamic economic dispatch with nonsmooth fuel cost functions. Proceedings of the World Congress on Nature and Biologically Inspired Computing, Dec. 9-11, IEEE Xplore Press, Coimbatore, India, pp: 152-157. DOI: 10.1109/NABIC.2009.5393632

Vazquez, R.A., 2011. Training spiking neural models using cuckoo search algorithm. Proceedings of the Congress of Evolutionary Computation, Jun. 5-8, IEEE Xplore Press, New Orleans, LA, USA, pp: 679-686. DOI: 10.1109/CEC.2011.5949684

Walton, S., O. Hassan, K. Morgan and M.R. Brown, 2011. Modified cuckoo search: A new gradient free optimisation algorithm. Chaos, Solitons Fractals, 44: 710-718. DOI: 10.1016/j.chaos.2011.06.004

Xin, B., J. Chen, J. Zhang, H. Fang and Z.H. Peng, 2011. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy. Tran. Syst. Man Cybernet. Part C, 42: 744-767. DOI: 10.1109/TSMCC.2011.2160941

Xin, B., J. Chen, Z. Peng and F. Pan, 2010. An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Sci. China Inform. Sci., 53: 980-989. DOI: 10.1007/978-3-642-29353-5_7

Yang, X. S. and S. Deb, 2009. Cuckoo search via Lévy flights. Proceedings of the World Congress on Nature and Biologically Inspired Computing, Dec. 9-11, IEEE Xplore Press, Coimbatore, India, pp: 210-214. DOI: 10.1109/NABIC.2009.5393690

Yang, X.S., 2011. Optimization Algorithms. In: Computational Optimization, Methods and Algorithms, Koziel, S. and X.S. Yang (Eds.), Springer, Berlin, ISBN-13: 978-3-642-20859-1 pp: 13-31.

Yang, X.S. and S. Deb, 2010. Engineering optimisation by cuckoo search. Int. J. Math. Model. Numerical Optimisat., 1: 330-343. DOI: 10.1504/IJMMNO.2010.035430

Yildiz, A.R., 2012. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. Int. J. Adv. Manufacturing Technol., 64: 55-61. DOI: 10.1007/s00170-012-4013-7