A Novel Parallel Search Technique for Optimization

Manaar Alam
Department of Computer Science and Engineering
Indian School of Mines
Dhanbad, Jharkhand, India.
Email: alam.manaar@gmail.com

Soumyajit Chatterjee
Department of Computer Science and Engineering
Indian School of Mines
Dhanbad, Jharkhand, India.
Email: sjituit@gmail.com

Haider Banka
Associate Professor
Department of Computer Science and Engineering
Indian School of Mines
Dhanbad, Jharkhand, India.
Email: haider.banka@gmail.com

***Abstract*—In this paper a novel parallel search technique for solving optimization problems is proposed. The novelty of the proposed algorithm is that it uses a very less number of tunable parameters as compared to most of the similarly established stochastic algorithms. In addition, a simple operator called rotate left and complement (RLC) is used in this method that helps to reach distinct points in the search space instead of reaching repeated points as done by conventional meta-heuristics. It also helps to enhance the chance of reaching towards optimal results and save time by not reaching repeated points. Another operator called flip operator is used to introduce variations within the search space as well. Performance of the proposed algorithm is tested on eight benchmark functions available in literature and the results are compared with some of the well-known optimization techniques which show the completeness of the proposed algorithm.**

Keywords—Tunable Parameters, Parallel Search, Rotate Left Complement Operator (RLC), Flip Operator, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Cuckoo Search (CS).

I. INTRODUCTION

In the past few years there has been quite an appreciable number of research works done on algorithms designed for solving optimization problems. Most of them have specially designated operators that perform a specific set of operations on the randomly drawn input from any probability distribution function (can be both continuous as well as discrete). But, most of these techniques have a prime concern regarding what are known as Tunable Parameters. These are actually a set of user-defined parameters that help the algorithm to provide the desired results, when a suitable value depending on the kind of problem is assigned to them. A very simple example can be the crossover or mutation probabilities for Genetic Algorithm [6] or the inertia factor in case of Particle Swarm Optimization [6]. Though these parameters are meant to enhance the performance of the algorithm but assigning them the perfect value (may vary for different problems) adds to the anxiety. The algorithm proposed here has very few tunable parameters in comparison to the previous ones, thus resolving the issue of assigning optimum values for too many user-defined parameters.

Another important, though very intricate, detail of any metaheuristic algorithm is that usually it searches for a specific portion of the search space rather than the entire one. It has been seen through continuous experimentation that most of the previously described meta-heuristic algorithms usually search certain points repeatedly and due to this most of them perform unnecessary computation over the same solutions obtained from the search space. Another issue which creeps in due to this is that these algorithms skip many points in the search space thus, taking more time to converge. The algorithm proposed here implements an exhaustive, though parallel, search technique which will consider all the points in the search space exhaustively thus resulting in decreased number of unnecessary computations and providing quicker convergence. Moreover, searching all the potential solution points result in more accurate results thus reducing the chances of getting erroneous results.

In the next section, a literary survey of some existing metaheuristics are provided. In the third the mathematical background of this algorithm is discussed and in the subsequent fourth and fifth sections the performance analysis on the benchmark functions[5] and the conclusion is provided.

II. PRELIMINARIES

This section discusses the basics and applications of some of the meta-heuristic algorithms like –

1) Genetic Algorithm: It is a nature inspired meta-heuristic algorithm that is inspired from the genetics and natural selection. The algorithm[1] uses artificial forms of chromosomes as random points in the search space along with the operators of crossover and mutation to gradually explore and exploit the search space. The algorithm uses various kinds of selection mechanism to choose the fittest individuals, the very basic being the Roulette Wheel Selection. The algorithm takes number of populations, number of generations (can be replaced by a stopping criterion), chromosome length, crossover and mutation probabilities as parameters from the user. The two most important advantages of GA are its power of parallelism and its capability to handle complex problems[6]. But in spite of these advantages there are some major limitations[7] of it like proper identification of the fitness function, correct tuning of the user-defined parameters, not a good identifier of local optimum etc.. In these years GA has been used in a wide variety of applications[7] like solving problems in

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

Robotics, Image Processing, Gaming, Real Time Systems, Job Scheduling etc..

2) Particle Swarm Optimization: Like GA it is also a nature inspired algorithm based on the behavior of particles and social interaction in a swarm. This algorithm[2] starts with a number of particles scattered randomly in the search space and updates the global best based on the position of the best particle. The algorithm takes the swarm size (analogous to population size), maximum number of generations (can be replaced by a stopping criterion), initial inertial factor, priorities of global evaluation and self-introspection as parameters from the user. Some important advantages[8] of PSO lies in its properties of quick convergence, less dependence on initial points, less sensitivity to the nature of objective function etc.. But like many other metaheuristic algorithms PSO has the major drawback of having no guarantee that it will generate optimal results. Moreover the stochastic characteristics of the algorithm at times may lead to several problems of real life[8]. Applications[9] of PSO range from Signal Processing, Biomedical, Communication Networks, Clustering and Classification, Combinatorial Optimization etc..

3) Cuckoo Search: Another nature inspired algorithm based on the brood parasitism of some species of Cuckoos. This algorithm[3] considers a number of randomly chosen cuckoos from the search space, who are in search of the best nest or the global optimum. A fraction of the nests are considered to be worst and those nests or solutions are improved using Lévy Flights. The algorithm takes number of cuckoos, maximum number of iterations (can be replaced by a stopping criterion), step-size, Lévy Flight constant and new nest probability as parameters from the user. Advantages[10] of Cuckoo Search are mostly due to its property of quick convergence and efficiency. The main disadvantage[11] of the algorithm is that the quicker convergence cannot be assured as the algorithm is based on random walks. Applications[10] of CS are widespread in various fields starting from Structural Design, Web Services, Software Engineering to Scheduling Problems.

III. Proposed Algorithm

The algorithm starts with a random initial point on the search space and proceeds with generating new set of solutions at each generation. The process is repeated for a predefined number of iterations. The main idea behind this algorithm is that a single point appears only once, unlike Genetic Algorithm (GA), which saves the number of unnecessary evaluations. Apart from this, the algorithm always finds unique points in the search space, thereby increasing the probability of getting good results.

The most challenging part of this algorithm is to find distinct points on the search space. Here an operator will be discussed which is used in the proposed algorithm to generate unique points in the search domain. Moreover, another operator will also be explained which is used to incorporate more variations within the solutions. These variations will ensure more number of unique points and thus will help in improving the performance of the algorithm. But before that, say, -

- *A binary string σ of length L is used to represent a value x in the search space. In other words,* $\sigma = (\beta_{L-1}\beta_{L-2}\beta_{L-3} \dots \beta_1\beta_0), \beta_i \in \{0,1\}, \forall i$ *and σ maps to a real value x in the search space.*

If the length of the binary string is L, then total number of such strings possible is 2^L. The objective is to find the string with the optimal value.

A. Rotate Left Complement (RLC) Operator

For generating distinct solution points, the Rotate Left Complement (RLC) Operator[4] has been used, which is defined as follows –

- *For a binary string of length L fetch the most significant bit, i.e., β_{L-1}, rotate the string to the left and put the complement of β_{L-1}, i.e., $\bar{\beta}_{L-1}$ in the least significant position.*

Thus, this operator provides a new binary string of length L from the old string and thus the local optima is avoided.

Now, with an example the working of this operator is explained below –

Suppose there lies a solution σ of length 4, i.e., $L = 4$, at any instant τ. Say, $\sigma = 1001$. On the application of RLC operator the string σ produces 0010.

If the RLC operator is applied successively on the newly generated strings, all the distinct strings those are generated from 1001 by this process are - 0010, 0101, 1011, 0110, 1101, 1010, 0100. Thus, the string of length 4 can generate 7 distinct strings.

Using RLC operator, in general at most $2L-1$ distinct binary strings can be generated from a binary string of length L. Thus at most $2L-1$ distinct search points can be explored, in a search space of 2^L, from a single search point.

The main property of RLC operator is that if a string σ_2 is generated from string σ_1 using RLC operator, then σ_1 and σ_2 will never be equal. The proof of this property is stated as follows –

Say the string σ_1 is $\beta_{L-1}\beta_{L-2}\beta_{L-3} \dots \beta_1\beta_0$. Then σ_2 is $\beta_{L-2}\beta_{L-3}\beta_{L-4} \dots \beta_0\bar{\beta}_{L-1}$. , where $\bar{\beta}_{L-1}$ denotes the complement of β_{L-1}. If σ_1 is equal to σ_2, then

$$\beta_{L-1} = \beta_{L-2}$$
$$\beta_{L-2} = \beta_{L-3}$$
$$\beta_{L-3} = \beta_{L-4}$$
$$\vdots$$
$$\beta_2 = \beta_1$$
$$\beta_1 = \beta_0$$
$$\beta_0 = \bar{\beta}_{L-1}$$

Which in turn satisfies $\beta_{L-1} = \bar{\beta}_{L-1}$, which is not true. Hence, σ_1 and σ_2 can never be equal.

There can be one situation when a string σ_3, generated from σ_1 by applying RLC operator twice, may be equal to σ_1. For example –

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

Let a string σ_1 be 101. The string σ_2 generated by applying first RLC operation is 010. Similarly, the string σ_3 generated from σ_2 by applying RLC again is 101.

To come out from this situation, the next, another operator is introduced for incorporating variations within the solutions.

B. Flip Operator

More variations within the solutions are ensured by this Flip operator, which is defined as follows -

- For a binary string of length L replace each β_i by its complement, i.e., $\bar{\beta}_i$, one by one to generate L distinct binary strings.

Thus, this operator provides L new binary strings of length L from the old string.

Now, with an example the working of this operator is explained below –

Suppose we have solution σ of length 4, i.e., $L = 4$, at any instant τ. Let $\sigma = 1001$. On the application of the Flip operator the string produces 4 different binary strings which are - $0001, 1101, 1011, 1000$. This operator generates 2 new strings in the search domain of 2^L and thereby adds more variations within the solutions.

Thus, using these operators the algorithm searches distinct points in the search space and exhaustively checking every potential point of solution.

The proposed algorithm is given as follows –

Algorithm Pseudo Code for Parallel Search

1. Generate a random string of length L and store it in α.
2. Generate a set Δ of $2L - 1$ distinct solutions by RLC operator from α.
3. Find best string from Δ.
4. If γ is better than α then exchange α and γ.
5. Generate a set Φ of L distinct solutions from α by Flip Operator.
6. Find best string from Φ and store it in γ.
7. If γ is better than α then exchange α and γ.
8. **repeat**
9. Generate a random string of length L and store it in γ.
10. Generate a set Δ of $2L - 1$ distinct solutions by RLC operator from γ.
11. Find best string from Δ and store it in γ.
12. If γ is better than α then exchange α and γ.
13. Generate a set Φ of L distinct solutions from α by Flip Operator.
14. Find best string γ from Φ.
15. If γ is better than α then exchange α and γ.
16. **until** *maximum number of generations is reached*
17. Post process Results.

IV. COMPARISON

In this section the proposed algorithm will be tested with the previously mentioned meta-heuristic algorithms.

The following table provides the mathematical forms of the various uni-modal benchmark functions used for the comparison[5]. The first column represents the codes for the various uni-modal benchmark functions , which can be referred to, in the subsequent tables. The next column depicts the actual name of the functions and the last column represents the mathematical formulation of the functions.

Table I

DIFFERENT BENCHMARK FUNCTIONS (UNIMODAL)

Code	Name	Function
AF	Ackley Function 2	$-200e^{-0.02\sqrt{x_1^2+x_2^2}}$
PF	Powell Sum Function	$\sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$
RF	Rotated Ellipse Function	$7x_1^2 - 6\sqrt{3}x_1x_2 + 13x_2^2$
ZF	Zirilli or Aluffi-Pentini's Function	$0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$

The following table depicts the values of each Benchmark functions with their search space and optimum values. Here in this table the first column depicts the code for the specific benchmark function. The next column specifies the search space. The last column with the data format $f(x, y) = z$ represents that the function f has the global optimum at the points (x, y) and its optimum value at that point is z.

Table II

ABOVE BENCHMARK FUNCTIONS WITH THEIR GLOBAL OPTIMUMS

Code	Search Domain	$f(x_*)$
AF	$-32 \leq x_i \leq 32$	$f(0,0) = -200$
PF	$-1 \leq x_i \leq 1$	$f(0,0) = 0$
RF	$-500 \leq x_i \leq 500$	$f(0,0) = 0$
ZF	$-10 \leq x_i \leq 10$	$f(-1.0465,0) \approx -0.3523$

The following table compares the solutions obtained by GA, PSO and CS with that of the proposed algorithm. The first column represents the name of the algorithm used. The next column represents the value of the variables at the global optimum, i.e., if $x_1 = t, x_2 = u$ and $f(x_*) = v$, then it represents that the algorithm has found the solution at point (t,u) for the function represented in the third and subsequent columns, by their codes, and the value at that point is v.

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

Table III

COMPARISON OF SOLUTIONS

		AF	PF	RF	ZF
GA	x_1	0.093841	0.30205	-0.79687	-1.06006
	x_2	0.093841	-0.15347	-0.30611	-0.08221
	$f(x_*)$	-199.46985	0.32560	3.12820	-0.34879
PSO	x_1	≈ 0	0	0	-1.04668
	x_2	≈ 0	≈ 0	0	≈ 0
	$f(x_*)$	-200	0	0	-0.35238
CS	x_1	≈ 0	≈ 0	0	-1.04668
	x_2	≈ 0	≈ 0	0	≈ 0
	$f(x_*)$	-200	0	0	-0.35238
PS	x_1	0	0	0	-1.04492
	x_2	0	≈ 0	0	≈ 0
	$f(x_*)$	-200	0	0	-0.35238

The Table III compares the number of iterations required by GA, PSO and CS with the proposed algorithm. Here, the first column represents the codes of the benchmark functions and the second and subsequent columns represent the number of iterations taken by the specified algorithm.

Table IV

COMPARISON OF NUMBER OF ITERATIONS REQUIRED

Code	Number of Iterations			
	GA	PSO	CS	PS
AF	10	94	463	41
PF	334	114	494	37
RF	4224	140	706	35
ZF	118108	53	251	30

For GA the population sizes are 2000, 25, 200 and 500 respectively. For PSO and CS it is 50 for each. For the Proposed Algorithm $L = 64$.

The following table provides the mathematical forms of the various multi-modal benchmark functions [5] for the comparison. The first column represents the codes for the various multi-modal benchmark functions, which can be referred to, in the subsequent tables. The next column depicts the actual name of the functions and the last column represents the mathematical formulation of the functions.

Table V

DIFFERENT BENCHMARK FUNCTIONS (MULTIMODAL)

Code	Name	Function
AD	Adjiman Function	$\cos(x_1)\sin(x_2) - \frac{x_1}{x_2^2+1}$
BC	Bartels Conn Function	$\lvert x_1^2 + x_2^2 + x_1x_2\rvert + \lvert\sin(x_1)\rvert + \lvert\cos(x_2)\rvert$
BF	Bohachevsky Function 1	$x_1^2 + 2x_1^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
CF	Cross-in-Tray Function	$-0.0001\left[\left\lvert\sin(x_1)\sin(x_2)e^{\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert + 1\right]^{0.1}$

The following table depicts the values of each Benchmark functions with their search space and optimum values. Here in this table the first column depicts the code for the specific benchmark function. The next column specifies the search space. The last column with the data format $f(x, y) = z$ represents that the function f has the global optimum at the points (x, y) and its optimum value at that point is z. Here, a function can have more than one global optima.

Table VI

ABOVE BENCHMARK FUNCTIONS WITH THEIR GLOBAL OPTIMUMS

Code	Search Domain	$f(x_*)$
AD	$-1 \le x_1 \le 2, -1 \le x_2 \le 1$	$f(2, 0.10578) = -2.02181$
BC	$-500 \le x_i \le 500$	$f(0,0) = 1$
BF	$-100 \le x_i \le 100$	$f(0,0) = 0$
CF	$-10 \le x_i \le 10$	$f(\pm 1.349406, \pm 1.349406) = -2.062612$

The following table compares the solutions obtained by GA, PSO and CS with that of the proposed algorithm. The first column represents the name of the algorithm used. The next column represents the value of the variables at the global optimum, i.e., if $x_1 = t$, $x_2 = u$ and $f(x_*) = v$, then it represents that the algorithm has found the solution at point (t, u) for the function represented in the third and subsequent columns, by their codes, and the value at that point is v. For the functions with more than one global optima, any one of the solutions is provided.

Table VII

COMPARISON OF SOLUTIONS

		AD	BC	BF	CF
GA	x_1	2	0.09742	0.03998	-1.36131
	x_2	0.05962	0.82034	-0.04301	-1.33755
	$f(x_*)$	-2.01771	1.54162	0.08336	-2.06257
PSO	x_1	2	≈ 0	≈ 0	-1.34941
	x_2	0.10578	≈ 0	≈ 0	1.34941
	$f(x_*)$	-2.02181	1	0	-2.06261
CS	x_1	2	≈ 0	≈ 0	1.34941
	x_2	0.10578	≈ 0	≈ 0	1.34941
	$f(x_*)$	-2.02181	1	0	-2.06261
PS	x_1	2	0	≈ 0	-1.34941
	x_2	0.10578	≈ 0	≈ 0	-1.34941
	$f(x_*)$	-2.02181	1	0	-2.06261

The following table compares the number of iterations required by GA, PSO and CS with the proposed algorithm. Here, the first column represents the codes of the benchmark functions and the second and subsequent columns represent the number of iterations taken by the specified algorithm.

978-1-4799-8579-1/16/$31.00 ©2016 IEEE

Table VIII

COMPARISON OF NUMBER OF ITERATIONS REQUIRED

Code	Number of Iterations			
	GA	PSO	CS	PS
AF	1037	51	156	30
PF	5870	102	404	36
RF	1980	58	277	27
ZF	1002	46	423	28

For GA the population sizes are 25, 200, 200 and 200 respectively. For PSO and CS it is 50 for each. For the Proposed Algorithm $L = 64$.

V. CONCLUSION

In this paper a novel algorithm has been introduced and a rotate left and complement operator (RLC) is used to generate distinct solutions and also to avoid unnecessary computations of the repeated points. The approach also introduced the flip operator, as well, which generates variations within the search space for improved results. The proposed algorithm has only two tunable parameters i.e., i) the string length (L), ii) the number of generations, which is much lesser in comparison to that of many other meta-heuristics. It is clearly eminent from the given results that the proposed algorithm can produce comparable and competitive result with less number of iterations.

ACKNOWLEDGMENT

The authors are thankful to Prof. C. A. Murthy, Indian Statistical Institute, Kolkata for his valuable contributions towards the paper.

REFERENCES

[1] Goldberg, D.E., Genetic Algorithms, Pearson Education, 2006.

[2] Russell Eberhart, James Kennedy, A New Optimizer Using Particle Swarm Theory, 1995.

[3] Xin-She Yang and Suash Deb, Cuckoo Search via L´evy Flights, In Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), India, IEEE Publications, USA, pp 210-214, 2009.

[4] B. Brey, Intel microprocessors : 8086/8088,80186,80286,80386, and 80486 architecture, programming, and interfacing., New Delhi: Prentice Hall, 1995.

[5] Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimisation problems, Int. J. Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, 2013.

[6] Xin-She Yang, Nature-Inspired Optimization Algorithms, Elsevier, First Edition, 2014.

[7] S.N. Sivanandam and S.N. Deepa, Introduction to Genetic Algorithms, Springer, 2008.

[8] Kwang Y. Lee and Jong-Bae Park, Application of Particle Swarm Optimization to Economic Dispatch Problem: Advantages and Disadvantages, IEEE, 2006.

[9] Riccardo Poli, Analysis of the Publications on the Applications of Particle Swarm Optimization, Journal of Artificial Evolution and Applications, Vol. 2008, Article No. 3, Jan 2008.

[10] Xin-She Yang, Cuckoo Search and Firefly Algorithm: Theory and Applications, Springer, 2014.

[11] S. Walton,O.Hassan, K. Morgan and M.R. Brown Modified cuckoo search: A new gradient free optimisation algorithm, Chaos, Solitons and Fractals, Elsevier, Vol. 44, Issue 9, pp- 710-718 Sept. 2011.

978-1-4799-8579-1/16/$31.00 ©2016 IEEE