IJARCCE

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 8, August 2021

DOI 10.17148/IJARCCE.2021.10807

Exploration and Exploitation Strategies in Jaya Algorithm: Short Review

Sandeep U. Mane[1], M. R. Narsingarao[2]

Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India[1]

Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India[2]

Abstract: The Jaya algorithm is one of the recently developed innovative, algorithm-specific parameter-less optimization algorithm. This study presents a short review of exploration and exploitation approaches used by researchers to improve the performance of the basic Jaya algorithm. The objective of this paper is to present collectively the important strategies adopted to modify the basic Jaya algorithm focusing on exploration and exploitation only. This study considers the recent journal publications about the Jaya algorithm and its improvement. It is observed that researchers have focused on the solution update equation of the basic Jaya algorithm to improve the performance and balancing the exploration and exploitation in the modified Jaya algorithm.

Keywords: Exploration, Exploitation, Jaya algorithm, Strategies to balance exploration and exploitation.

I. Introduction

The last few decades have experienced massive development in the field of optimization algorithms. Various researchers have proposed different types of algorithms to solve single, bi-objective or multi-objective optimization problems. Nature-inspired algorithms are proposed by observing and mimicking natural phenomena. These algorithms are classified as evolutionary and swarm-intelligence based algorithms. The nature-inspired algorithms are probabilistic it requires the fine-tuning of the algorithm's explicit parameters to obtain better results. Rao has developed two algorithms namely TLBO and Jaya, which are not bounded with algorithm-specific parameters. These two algorithms work fine with common controlling parameters [1].

The success of heuristic or nature-inspired algorithms depends on, how it balances the exploration and exploitation during searching of solution. The algorithm gives better results when these two operations are performed efficiently. Exploration is the process that explores or discover the new solution regions and maintains diversity among the individuals of the solution set. The exploitation is searching for the solution in the neighbourhood of the current best solution. If algorithms focus is on exploration, the convergence rate will get affected, while during exploitation the algorithm may get trapped into local optima. The exploration phase though maintains diversity among individuals, it leads to random search and obtaining the global best solution becomes a tedious task. Too much focus on exploitation fails to maintain diversity in the solution and algorithms get quickly converged [2].

The behaviour of any population-based algorithm is largely controlled by its exploration and exploitation ability at the time of progression [3]. The controlling and measuring of exploitation and exploration in evolutionary algorithms (EAs) is too hard. If the researcher could able to understand the exploration and exploitation very well then it is possible to develop better heuristic algorithms as well as it will help to reduce computational efforts. The exploration and exploitation are controlled by using various operators and parameters in evolutionary algorithms. The crossover and mutation operator controls the exploration while selection maintains the exploitation in evolutionary algorithms [2]. Črepinšek et al. have presented a literature survey on exploration and exploitation in evolutionary algorithms. Authors have discussed how exploration and exploitation are achieved in EAs, the situation where it is to be controlled and different approaches found in the literature to balance the exploration and exploitation [2].

The swarm-based algorithms make use of various algorithm-specific parameters to perform exploration and exploitation. The inertia weight, social and cognitive parameters used by Particle swarm optimization algorithm, the number of an onlooker, employed, and scout bees required to set in Artificial bee colony algorithm, while Ant colony optimization uses pheromone evaporation rate. The accurate tweaking of such operators is a vital task to balance the exploration and exploitation in nature-inspired algorithms [4].

This study presents the basic Jaya algorithm, the issues raised by researchers with the Jaya algorithm to perform exploration and exploitation, and various strategies devised by researchers to overcome these issues. The purpose of this review is not to criticize the author of this algorithm but help other researchers to advance their work using the Jaya algorithm. The motive behind this study is to present a review of different strategies developed by researchers for improving the performance of the Jaya algorithm to solve different optimization problems. This study will help other researchers to choose appropriate exploration and exploitation strategies for their problems. The articles selected for the study are from reputed journals and the search is performed as “exploration and exploitation in Jaya algorithm” in Google scholar.

The remaining paper is organized as, Section II briefs about the basic Jaya algorithm, different exploration and exploitation strategies used to modify the basic Jaya algorithm are presented in Section III, and finally, Section IV presents the discussion and concludes the findings.

II. Basic Jaya Algorithm

Rao has developed a novel, population-based metaheuristic algorithm - Jaya to solve optimization problems from various domains [1]. The Jaya algorithm is an algorithm that is unrestricted from fine-tuning of algorithm-specific operators. This feature reduces the complexity as well as the computational efforts of application developers or researchers. The initial randomly generated solutions are updated in the next iterations using Eq. 1.

$$A(i+1, j, k) = A(i, j, k) + r(i, j, 1)(A(i, j, b) - |A(i, j, k)|) - r(i, j, 2)(A(i, j, w) - |A(i, j, k)|) \quad (1)$$

Eq. 1 consists of solutions from the previous iteration, best and worst solutions from the current iteration, and random numbers. The best and worst individual from the previous iteration is scaled using two random numbers to guarantee good diversified solutions. These random numbers have an impact on exploring the feasible region. The exploitation is achieved using the term |A (i,j,k)| in Eq. 1.

Basic Jaya algorithm’s steps are presented below.
1. Initialize population, design variable, and set total iterations.
2. Evaluate the Objective Function.
3. Identify the best and worst individuals in the generated population.
4. Modify the other individuals based on the best and worst individuals using Eq. (1).
5. Compare the new individuals with the previous and continue with the best individuals and skip the worst individuals.
6. If stopping criteria are not satisfied then go to step 3.
7. If the stopping criterion is satisfied then halt and print the optimal solution.

The detailed working of the Jaya algorithm and demonstration of the Jaya algorithm to solve single-objective constrained and unconstrained optimization problems are found in [1].

Different researchers have modified and developed improved versions of the Jaya algorithm to solve single, multi-objective, constrained, and unconstrained optimization problems found in various domains. Rao has provided the updated list of research works published using the Jaya algorithm on his webpage and can be found at https://sites.google.com/site/jayaalgorithm. (The authors of this paper have last accessed the webpage on 15 April 2020, 10.00 am IST).

Exploration and exploitation are the two major components of any nature-inspired or heuristic algorithms. The balanced exploration and exploitation searching behaviour of an algorithm leads to obtain the optimum solutions for selected optimization problems with optimum computational efforts [5].

The developing philosophy behind the Jaya algorithm is to push the fitness function value towards the solution space which contains an improved value for the fitness function than earlier iteration. The Jaya algorithm tries to grow towards the finest possible value of a fitness function, so the exploitation will be more dominating [6].

Though the Jaya algorithm is simple to implement as well as efficient, it is trapped in local minima, less diversified solutions, and weak exploration ability. The individuals in a population in the Jaya algorithm are mainly updated by best and worst values from earlier iteration. This leads to premature convergence and affects the accuracy of solutions [5]. The Jaya algorithm is weak in social communication among the individuals in the population; it leads to weak exploration behaviour of the algorithm. The basic Jaya algorithm does not have any mechanism or strategy to come out of the local optima or to control the random exploration [7].

Researchers have suggested different mechanisms to be incorporated into the Jaya algorithm for improving and achieving the balance between exploration and exploitation in the basic Jaya algorithm. These mechanisms are introduced to solve specific optimization problems.

III. Different Exploration and Exploitation Strategies Used to Modify Jaya Algorithm

This section presents the overview of different mechanisms developed by researchers for balancing the exploration and exploitation behaviour of the basic Jaya algorithm. The strategies developed or used by researchers helps to avoid trapping in local optima as well as avoids random exploration and preserves the diversity among individuals in the population.

- **Levy flight (LF) and Greedy selection**

Ingle and Jatoth in [6] used Levy flight (LF) and a Greedy selection scheme to improve the performance of the basic Jaya algorithm. The proposed approach is JAYALF. The LF concept improves the diversity in generated populations and helps to avoid the non-improvement in the solution. The greedy selection method improves the exploitation of an algorithm without affecting the population diversity. Exploration and exploitation are balanced by incorporating the adaptive Levy index. The Levy flight are various natural and artificial entities such as earthquake analysis, fluid dynamics, cooling behaviour, the diffusion of fluorescent molecules, noise etc. and it's step length is measured from its distribution. The LF redistributes the obtained solutions from the current iteration around the solution space to avoid the loss of diversity in population, and in turn, discover the new feasible solution region. The LF helps the Jaya algorithm to come out from local optima if it is stuck. Greedy selection strategy from Differential Evolution (DE) algorithm improves the exploitation of the Jaya algorithm without affecting the population diversity. The use of a Greedy selection scheme incorporates the survival of fittest and preserves the best (elite) solution from each iteration. The best solution is obtained using the Jaya algorithm and sorted from best to worst. The 50% best solutions are then modified with the LF index. Greedy selection is performed to select the best solutions from the solutions before modification and after modification with LF. The process continues until the matching of termination criteria. The proposed approach is tested for the non-linear channel equalization problem [6].

- **Self-adaptive weight, Experience-based learning strategy, and Chaotic elite learning**

Yu et al. in [8] developed an Improved Jaya algorithm. The self-adaptive weight is used to improve the exploitation operation. The experience-based learning strategy i.e. the experience of other solutions is used to develop the learning strategy. It improves population diversity by exploring the feasible region. The chaotic elite learning approach avoids the trapping of the searching process in local optima in case of multi-modal problems. The authors have tested the proposed approach for identifying parameters of the photovoltaic model [8].

- **Selection and use of 3 best solutions**

The three (3) best solutions (best1, best2, and best 3) were selected from the population. The solution update equation from the basic Jaya algorithm has one change only; instead of single best, randomly one individual is selected from available three best individuals and used to generate the new individuals. This approach was proposed by Oclo et al. to improve the exploration. The proposed strategy also helps to maintain diversity in the population [9].

- **Multi-team perturbation guiding principle**

Rao and Keesari in [10] modified the Jaya algorithm to improve the searching using multiple teams. Authors have used multiple teams with the same population and each team has separate equations for driving the individuals to optimal region. Each team updates the population by using the movement equation, if any team’s movement equation does not return a good result, then that equation will be updated online by the leading solutions generated by other teams. This approach helps to improve exploration. To avoid trapping into local optima, the non-improved individuals are selected with a certain probability and such solution replaces the higher quality solutions. It increases diversity and explores the search space. Authors have used the following movement equations to modify the basic Jaya algorithm [10].

1) Perturbation equation-I

The perturbation equation (Eq. 2), the authors have taken as it is from the basic Jaya algorithm.

$$u'v,p,i = uv,p,i + r1,v,i\left(uv,best,i - |uv,p,i|\right) - r2,v,i\left(uv,worst,i - |uv,p,i|\right) \tag{2}$$

2) Perturbation equation-II

The second perturbation equation (Eq. 3) uses a chaotic random number to replace the random numbers r1and r2.

$$\begin{gathered} cm+1 = 4cm\,(1 - cm) \\ u'v,p,i = uv,p,i + cm,v,i\left(uv,best,i - |uv,p,i|\right) - cm,v,i\left(uv,worst,i - |uv,p,i|\right) \end{gathered} \tag{3}$$

3) Perturbation equation-III

The movement equation (Eq. 4) was developed using the concept of 'opposition-based learning'. Authors have used this equation earlier to propose the 'Quasi-oppositional-based Jaya algorithm (QO-Jaya)'. The quasi-opposite value of an individual is given by Eq. 4,

$$\begin{gathered} u^q_{v,p,i} = rand\,(a,\ b) \\ \text{where} \\ a = (LB_v + UB_v)/2 \\ b = LB_v + UB_v - uv,p,i \end{gathered} \tag{4}$$

4) Perturbation equation-IV

The movement equation-IV (Eq. 5) is taken from [9]. The movement equation is as follows:

$$u'v,p,i = uv,p,i + r1,v,i\left(uv,best(rb),i - |uv,p,i|\right) - r2,v,i\left(uv,worst,i - |uv,p,i|\right) \tag{5}$$

5) Perturbation equation-V

This movement equation (Eq. 6) is developed using the random chaotic number and it is taken from [8].

$$u'v,p,i = uv,best,i + rand\,(2cm - 1) \tag{6}$$

6) Perturbation equation-VI

This movement equation (Eq. 7) is developed using movement equation-V. The random generated between [0, 1] replaces the chaotic random number.

$$u'v,p,i = uv,best,i + rand\,(2rand - 1) \tag{7}$$

- **Adaptive multi-team perturbation guiding principle**

Rao et al. in [11] proposed an Adaptive Multi-Team Perturbation Guiding mechanism to improve the Jaya algorithm. The movement equations used in [10] is used here. The searching is performed using a single set of populations. The selected movement equation guides the population in different feasible areas of the search space. The function evaluation value is used to decide the count of teams. This approach reduces the pressure of exploring the entire search space. The adaptive number of team's selection affects both exploration and exploitation [11].

- **A linear decreasing inertia weight, Neighborhood search, and Use of elitism**

Raut and Mishra in [12] modified the basic Jaya algorithm for balancing between local and global search, i.e. exploitation and exploration. The authors have modified the solution update equation, from the basic Jaya algorithm (Eq. 8).

$$\begin{gathered} Y^{new}_{m,j,k} = s(t)\times Y_{m,j,k} + r\times(Y_{m,best,k} - |Y_{m,j,k}|) - r1\times(Y_{m,worst,k} - |Y_{m,j,k}|) \\ s(t) = s_{\max} + (s_{\max} - s_{\min})\times\frac{itr}{itr_{\max}} \end{gathered} \tag{8}$$

Where the s_{max} and s_{min} present the max and min value of inertia weight (s) and itr and itr_{max} are present and max iteration number respectively.

The large value of inertia weight during earlier iterations facilitates the exploration. The inertia weight value is slowly deceased after each iteration which performs the exploitation of the algorithm. To avoid local trapping, authors have introduced neighbourhood search. Also, it helps to maintain diversity among individuals. The elitism is incorporated to preserve the best individual from the current iteration and use it in the next iteration. It will also avoid the local trapping of the algorithm.

- **Use of DE crossover and the Cauchy mutation operator**

Wu and He in [5] used the crossover operator and Cauchy mutation operator from the Differential Evolution (DE) for improving the exploration and exploitation capability of the Jaya algorithm. The developed approach is a hybrid approach to improve the capability of the Jaya algorithm. The crossover operator helps to explore the search space while the Cauchy mutation operator improves the exploitation of the basic Jaya algorithm. The researchers who propose the hybrid Jaya approach needs to set the algorithm-specific parameters of the selected heuristic algorithm to hybridize with the Jaya algorithm. The appropriate tuning of the algorithm-specific parameter can be a limitation of such approaches.

- **Use of one-step K-means clustering and chaotic sequence**

In [13] Chen et al. modified the basic Jaya algorithm to balance between diversification and intensification. The one-step K-means clustering is incorporated before generating the new individuals. The initial population is divided into K – clusters. The minimum number of clusters is 2 and the maximum is the total number of individuals in the population. The K-means clustering works like a crossover operator. The chaotic sequence is applied after generating new solutions. The chaotic sequence restricts the algorithm trapping into local optima.

- **Parallel execution of Jaya and Bat algorithm – Hybrid approach**

Kaur et al. in [14] have presented a hybrid Jaya-Bat approach to improve the exploration and exploitation to obtain better optimum value. The Jaya and Bat algorithm works in parallel to obtain the best optimum value. If the Jaya algorithm gets stuck at local optima and fails to explore the search space to obtain globally best value then the Bat algorithms best and worst individual will help the Jaya algorithm to come out from the local convergence. Also, if the Bat algorithm gets stuck into local optima, the Jaya algorithm will help to escape from it. This hybrid approach balances between exploration of Jaya and exploitation of the Bat algorithm.

- **Use of Scaling factor, Multi-start adaptive, and elitism scheme**

To augment the exploration and exploitation in the Jaya algorithm to solve the multi-objective optimization problem, Zamli et al. used a multi-start adaptive scheme and elitism [7]. The difference between the term $r_1(X_{best} - |X_i^{(t)}|)$ and $r_2(X_{worst} - |X_i^{(t)}|)$, if it is large then the Jaya algorithm explores the search space and if it is small then it may get trapped into local optima. To balance between exploration and exploitation this difference should large in earlier iterations and it should steadily decrease as the iteration number increases. The authors have used the scaling factor to scale the $r_1(X_{best} - |X_i^{(t)}|) - r_2(X_{worst} - |X_i^{(t)}|)$ term.

Initially, the scaling factor was large and it was reduced as the iteration number increases, so it facilitates both exploration and exploitation. To get out from local optima, the authors have introduced a multi-start scheme in the modified Jaya algorithm. Restarting the whole search process of the algorithm, allows the algorithm to explore the search space again. To ensure enough exploration, the elitism scheme is used by authors.

- **Two group adaption strategy**

The two-group adaption scheme is used to propose the enhanced Jaya algorithm in [15]. The best and worst individual from the previous iteration affects the exploration and exploitation in the current iteration of the Jaya algorithm's update equation. In the proposed approach author have grouped the best individual and worst individual in two groups and computed the mean value of each group. That mean value is used to generate new individuals. The proposed E-Jaya algorithm considers the behaviour of all the individuals, which are the better and the worse individuals.

- **Performance-guided evolution process**

Yu et al. have modified the basic Jaya algorithm by utilizing the individual's performance to improve the exploration and exploitation capability while balancing the same [8]. Initially, all the individuals from the solution set are sorted in ascending order. Each ranked individuals' probability is computed. Each individual self-adaptively selects the solution update strategy based on its probability. The better individuals will explore the search space while the poorer will focus on exploitation. The authors have used two solution update strategies. If the individual has a better probability value it selects the strategy - II for improving the exploration otherwise the strategy - I selected for improving the exploitation ability.

The strategy - I make use of a self-adaptive weighting scheme, as given in Eq. 9:

$$x'_{i,j} = x_{i,j} + rand_1\left(x_{best,j} - |x_{i,j}|\right) - w \cdot rand_2\left(x_{worst,j} - |x_{i,j}|\right)$$
$$w = \begin{cases} \left(\dfrac{f(x_{best})}{f(x_{worst})}\right)^2, \text{if} \quad f(x_{worst}) \neq 0 \\ 1, \text{otherwise} \end{cases} \tag{9}$$

Strategy - II selects two individuals randomly (other than best and worst) to decide the searching direction for exploring the different feasible regions for improving the solution quality. The individuals are selected based on the probability value. The new individual is updated as per Eq. 10.

$$x'_{i,j} = x_{i,j} + rand \cdot \left(x_{i,j}^{Pl} - x_{m,j}^{R}\right) \tag{10}$$

$x_{i,j}^{Pl}$ and $x_{m,j}^{R}$ - The values of the jth variable for the l and m individuals respectively, the rand is a random number generated between [0, 1].

- **Use of XOR logic gate operation**

Aslan et al. solved the binary optimization problem using a modified Jaya algorithm, JayaX [16]. The authors have used the XOR logic gate to develop the JayaX algorithm. The JayaX algorithm preserves the properties of the basic Jaya algorithm, as free from algorithm-specific parameters. The XOR operation maintains the balance between exploration and exploitation. The individuals in the next generation are obtained using the modified solution update equation [16]. It is given as in Eq. 11,

$$x'_{k,j} = \left\{x_{k,j} \oplus \left(Best_j \oplus Worst_j\right)\right. \tag{11}$$

- **Chaotic based mutation strategy**

Farah and Belazi studied the basic Jaya algorithm and proposed the chaotic based Jaya algorithm to solve unconstrained numerical optimization problems in [17]. The balance between exploitation and exploration is important in any heuristic algorithm. The extreme exploitation will perform only local search while extreme exploration will perform a random search and the algorithm cannot give the optimum solution. The authors have modified the solution improvement equation from the original Jaya algorithm and proposed three mutation-based equations. These are given as,

$$Xnew_{i,j} = chaos_{i,j}Xrand_{i,j} + chaos_{i,j}\left(X_{i,j} - chaos_{i,j}Xrand_{i,j}\right) + chaos_{i,j}\left(Xbest_j - chaos_{i,j}Xrand_{i,j}\right) \tag{12}$$

$$Xnew_{i,j} = chaos_{i,j}Xrand_{i,j} + chaos_{i,j}\left(X_{i,j} - chaos_{i,j}Xrand_{i,j}\right) + chaos_{i,j}\left(Xworst_j - chaos_{i,j}Xrand_{i,j}\right) \tag{13}$$

$$Xnew_{i,j} = chaos_{i,j}Xbest_j + chaos_{i,j}\left(Xrand_{i,j} - S_F Xbest_j\right) \tag{14}$$

$chaos_{i,j}$ = absolute value of a chaotic variable.

SF = scaling factor.

Eq. 12 is useful for large-scale problems. It improves the exploration capability of the algorithm and increases the diversity among individuals in the population. This equation is not suitable when the problem contains a huge local optimum. Eq. 13 improves the global searching ability and population diversity. This equation avoids the worst solutions. Eq. 14 is strong in performing exploitation and improves convergence. The chaotic value and random numbers are used to select the above equations on a random basis at each iteration.

- **Combining solutions from previous and current iterations and sorting**

Recently Many-objective Jaya algorithm is proposed by introducing the non-dominated sorting, ranking and selection operator from the NSGA-II algorithm [18]. Authors have combined the population from current and previous iterations for improving and balancing the exploration and exploitation. After sorting the solutions best and worst individuals were selected from the entire population to generate the solution in the next iteration. In this approach, the selection operator is used to select only the best individuals from the sorted list, equal to population size.

IV. Conclusion

The Jaya algorithm is one of the algorithm-specific parameter less population-based approaches. It is a simple and proven technique to solve constrained and unconstrained optimization problems. The Jaya algorithm and its variations exist in the literature to solve single and multi/many-objective optimization problems from various domains [19. After reviewing the literature, it is found that researchers have identified that the Jaya algorithm has scope for improvement by modifying certain steps, provided that the basic feature of Jaya – free from algorithm-specific parameters is to be preserved.

The researchers have mainly considered the solution update equation from the basic Jaya algorithm and it is modified. The modification in the solution update equation has helped to improve the performance of the basic Jaya algorithm. The exploration, as well as exploitation capability, is improved due to modification. The improved Jaya algorithm prevented trapping into local optima. The diversity among individuals in the population is improved. Some of the researchers have proposed the hybrid version of the Jaya algorithm to improve its performance.

From this study, it is observed that researchers have considered only the solution update equation to introduce modifications. The different choices used for equation modification are the best individual in population set, random chaotic number, adaptive levy index, inertia weight, crossover and mutation operator from DE algorithm, probabilistic approach, XOR logic operation, etc. The purpose of this study is to present collectively the strategies adopted by researchers to improve the balance between exploration and exploitation in the Jaya algorithm.

In future work, different evolutionary and swarm operators can be introduced to improve the performance of the Jaya algorithm. Also, the Jaya algorithm can be hybridized with other nature-inspired algorithms. The researchers can do the modifications in such a way that, it should not disturb the beauty of the Jaya algorithm, i.e. parameter-less. The hybridization should not increase the computational efforts of the developer to tune any algorithm-specific parameters. The modifications can be introduced by considering the application or nature of the problem.

References

[1]. R. Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems”, International Journal of Industrial Engineering Computations, Vol. 7, No. 1, pp. 19-34, 2016.

[2]. M. Črepinšek, S. H. Liu and M. Mernik, “Exploration and exploitation in evolutionary algorithms: A survey”, ACM computing surveys (CSUR), Vol. 45, No. 3, pp. 1-33, 2013.

[3]. K. Yu, B. Qu, C. Yue, S. Ge, X. Chen, and J. Liang, “A performance-guided JAYA algorithm for parameters identification of photovoltaic cell and module”, Applied energy, 237, pp. 241-257, 2019.

[4]. Rao, S. S, “Engineering optimization: theory and practice”, John Wiley & Sons, 2019.

[5]. Wu, C., & He, Y. (2020). Solving the set-union knapsack problem by a novel hybrid Jaya algorithm. Soft Computing, 24(3), 1883-1902.

[6]. Ingle, K. K., & Jatoth, R. K. (2020). An Efficient JAYA Algorithm with Lévy Flight for Non-linear Channel Equalization. Expert Systems with Applications, 145, 112970.

[7]. Zamli, K. Z., Alsewari, A., & Ahmed, B. S. (2018). Multi-Start Jaya Algorithm for Software Module Clustering Problem. Azerbaijan Journal of High Performance Computing, 1(1), 87-112.

[8]. Yu, K., Liang, J. J., Qu, B. Y., Chen, X., & Wang, H. (2017). Parameters identification of photovoltaic models using an improved JAYA optimization algorithm. Energy Conversion and Management, 150, 742-753.

[9]. Ocłoń, P., Cisek, P., Rerak, M., Taler, D., Rao, R. V., Vallati, A., & Pilarczyk, M. (2018). Thermal performance optimization of the underground power cable system by using a modified Jaya algorithm. International Journal of Thermal Sciences, 123, 162-180.
[10]. Rao, R. V., & Keesari, H. S. (2018). Multi-team perturbation guiding Jaya algorithm for optimization of wind farm layout. Applied Soft Computing, 71, 800-815.
[11]. Rao, R. V., Keesari, H. S., Oclon, P., & Taler, J. (2019). Improved multi-objective Jaya optimization algorithm for a solar dish Stirling engine. Journal of Renewable and Sustainable Energy, 11(2), 025903.
[12]. Raut, U., & Mishra, S. (2019). An improved Elitist–Jaya algorithm for simultaneous network reconfiguration and DG allocation in power distribution systems. Renewable Energy Focus, 30, 92-106.
[13]. Chen, F., Ding, Z., Lu, Z., & Zeng, X. (2018). Parameters identification for chaotic systems based on a modified Jaya algorithm. Nonlinear Dynamics, 94(4), 2307-2326.
[14]. Kaur, A., Sharma, S., & Mishra, A. (2019). A Novel Jaya-BAT Algorithm Based Power Consumption Minimization in Cognitive Radio Network. Wireless Personal Communications, 108(4), 2059-2075.
[15]. Gong, C. (2017). An Enhanced Jaya Algorithm with a Two Group Adaption. International Journal of Computational Intelligence Systems, 10(1), 1102-1115.
[16]. Aslan, M., Gunduz, M., & Kiran, M. S. (2019). JayaX: Jaya algorithm with xor operator for binary optimization. Applied Soft Computing, 82, 105576.
[17]. Farah, A., & Belazi, A. (2018). A novel chaotic Jaya algorithm for unconstrained numerical optimization. Nonlinear Dynamics, 93(3), 1451-1480.
[18]. Mane, S., Narsingrao, M., & Patil, V. (2018). A many-objective Jaya algorithm for many-objective optimization problems. Decision Science Letters, 7(4), 567-582.
[19]. Rao, R. V. (2019). Jaya optimization algorithm and its variants. In Jaya: An Advanced Optimization Algorithm and its Engineering Applications (pp. 9-58). Springer, Cham.

BIOGRAPHY

Sandeep U. Mane is a research scholar at K L Deemed to be University, Guntur district AP, India and presently working as an Assistant Professor in Computer Science and Engineering department at Rajarambapu Institute of Technology, Sakhrale, MS, India. He has received a Bachelor of Engineering in Information Technology from BVCOE, Kolhapur, MS, India, and a Master of Technology (M. Tech.) in Computer Engineering from Dr. BATU, Lonere, MS, India. His research interests include multi-objective and many-objective optimization evolutionary algorithms and problems, High-Performance Computing using GPGPU, Parallelization of Heuristic and Meta-Heuristic Methods. He has published about 25 research papers in International Journals and Conferences.

M. R. Narsing Rao is presently working as a Professor in Computer Science and Engineering department at KLEF K L Deemed to be University, Guntur district AP, India. He has received his PhD degree in Computer Science and Systems Engineering from Andhra University, Visakhapatnam, 2012, Andhra Pradesh, India. He has more than 17 years of teaching experience at UG and PG. His research interests include Applications of Neural Networks, Content-Based Information Retrieval, Automata Theory, Bio-informatics, and Software Engineering. He has published more than 32 research papers in International and National Conferences and International Journals.