International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Hybrid Jaya Algorithm for Solving Multi-Objective 0-1 Integer Programming Problem

Surbhi Tilva, Jayesh Dhodiya

Abstract**:** ***The aim of this paper is to find the optimal solution of complex multi-objective 0-1 integer programming problem(IPP) where as other evolutionary approaches are fails to achieve optimal solution or it may take huge efforts for computation. This paper presents the Hybrid Jaya algorithm for solving Multi-objective 0-1 IPP with the use of exponential membership function. In this work, we have improved the Jaya algorithm by bring in the conception of binary and exponential membership function. To established the effectualness of the suggested algorithm, one mathematical illustration is given with a data set from the practical and sensible state. At the end, the response of the improved algorithm is compared with other reported algorithms and we found that the suggested algorithm is evenly good or better for obtaining the solution of multi-objective 0-1 IPP.***

Index Terms**:** ***α-Level sets, Exponential membership function, Jaya algorithm, Multi-Objective 0-1 Integer Programming Problem.***

I. INTRODUCTION

An optimization problem fundamentally deals with searching the best solution from all the feasible solutions and it0020is known as optimal solution. An important part of research and application in the area of optimization are taken as a single objective while the almost practical tasks considers two or more than two objectives. The existence of several contradictory or clashing objectives is normal in lots of problems and which creates the optimization problems further fascinating to solve.

Multi-objective optimization problems (MOOP) are solved using two kinds of approaches namely classical and evolutionary approaches. The classical approaches find only single best solution in one simulation run, therefor those approaches are problematic to deal with MOOP while in the evolutionary approaches we can search multiple optimal solutions in one simulation run because of their population approach. The acknowledged evolutionary approaches are: Genetic Algorithm, Evolution Strategy, Artificial Immune Algorithm, etc. Most of the evolutionary approaches require algorithm-specific parameters that play crucial role in obtaining the optimal solution and defining these algorithm-specific parameters is a difficult task that can lead to extra computational efforts. So, this fact motivated R. V. Rao to introduce the TLBO algorithm which doesn't need any kind of algorithm-specific parameters [1]. It mimics the teaching-learning process.

In 2016, R. V. Rao introduced another algorithm called Jaya algorithm (JA) which is also independent of the controlling parameters [2]. This algorithm has been widely accepted as compare to the TLBO algorithm because TLBO algorithm works on dual stages i.e. teaching stage and learning stage while Jaya algorithm works only on a single stage.

The Jaya algorithm has been used by numerous authors to solve different kinds of problems from various domains. Kumar Abhishek, V. Rakesh Kumar, S. Datta, and S. S. Mahapatra find the solution for selecting the optimal process parameter configurations while the turning of CFRC by integrating the nonlinear regression model and FIS with the JA [3]. W. Warid, H. Hizam, N. Mariun, and N. Abdul-Wahab proposed the meta-heuristic optimization approach to solve the OPF problem using JA [4]. S. P. Singh, T. Prakash, V. P. Singh, and M. G. Babu have been proposed the JA based PID controller for AGC, which interconnects power system in two areas [5]. S. Phulambrikar used JA to solve CEED problem [6]. R. V. Rao and G. G. Waghmare investigate the performance and efficiency of JA to solve the complex constrained design optimization problem [7]. S. Mishra and P. Ray optimized the filter parameters and integral controller of PV-DSTATCOM by JA [8]. There are also a few variations of Jaya algorithms that are found in the literature. These variations are Multi-objective Jaya algorithm (MOJA) [9], Multi-objective Quasi-oppositional Jaya algorithm (MOQOJA) [10], Binary Jaya algorithm (BJA) [11], Improved Jaya Algorithm (IJA) [12], elitist Jaya algorithm (EJA) [13], and Self-Adaptive Multi-Population Jaya algorithm (SAMPJA) [14]. Also the Jaya algorithms posterior version is used to solve MOOPs in simulation of a single run [9]. T. Prakash, V. P. Singh, S. P. Singh and S. R. Mohanty used the BJA to solve the optimal placement of PMU [11]. In BJA, the candidate solutions are represented in binary form (0,1). The improved gradient based Jaya algorithm for solving MOPs has been proposed by R. Azizipanah-Abarghooee, P. Dehghanian and V. Terzija [12]. R. V. Rao and A. Saroj introduced new elitist JA for obtain the solution of MOOPs [13]. R.

Revised Manuscript Received on December 30, 2019.

* Correspondence Author
Surbhi Tilva*, Applied Maths & Humanities Department, Sardar Vallabhbhai National Institute of Technology Surat, India.
Jayesh Dhodiya, Applied Maths & Humanities Department, Sardar Vallabhbhai National Institute of Technology Surat, India.

V. Rao and A. Saroj proposed new multi-population Jaya algorithm known as SAMPJA which has multi-population and self-adaptive strategy to obtain the solution of various unconstrained and constrained engineering optimization problems [14].

Now a days, there has been more concern for hybridization algorithm which is the combination of original optimization algorithm with some another concepts to improve its quality. The outcome of the obtained algorithm mostly always gives the batter results than the original one. As JA is an evolutionary approach, sometime it fails to achieve feasible optimum solution for higher variables. So, to achieve our aim we modified the basic JA into hybrid jaya algorithm.

II. PRELIMINARIES

A. Membership function:

A fuzzy set $P(y)$ is constitute of two components: (1) The element 'y' and (2) Its membership function '$\mu_P(y)$' and denoted as $P(y)=\{(y,\mu_P(y)),y\in Y;\mu_P(y)\in[0,1]\}$. The membership function (MF) gives a curve which is obtained by mapping each point of the input space to a membership value between 0 and 1. Let Z_k^L and Z_k^U be the lower and upper bound of objective Z_k respectively, then the exponential membership function is describe as follows:

$$\mu^E_{Z_k}(y)=\begin{cases}1, & \text{if } \mathrm{Z_k \le Z_k^L}.\\ \dfrac{e^{-s\psi_k(y)}-e^{-s}}{1-e^{-s}}, & \text{if } \mathrm{Z_k^L < Z_k < Z_k^U}.\\ 0, & \text{if } \mathrm{Z_k \ge Z_k^U},\ \forall \mathrm{k}.\end{cases}$$

Where, $\psi_k(y)=\dfrac{Z_k-Z_k^L}{Z_k^U-Z_k^L}, k=1,2,\cdots,m$ and s is known as shape parameter which is always non-zero and When $s>0(s<0)$ then, the MF is concave (convex) in $[Z_k^L,Z_k^U]$.

B. α-CUT:

Let P be a fuzzy set in Y and $\alpha\in(0,1]$. The α-cut of fuzzy set P is the crisp set P_α which is given as $P_\alpha=\{y\in Y:\mu_P(y)\ge\alpha\}$.

From the definition of α-cut, for any fuzzy set P and pair $\alpha_a,\alpha_b\in(0,1],\alpha_a\le\alpha_b$, we have $P_{\alpha_b}\subseteq P_{\alpha_a}$. So, every α-cuts of whatever fuzzy set from families of crisp sets can be used to present a given fuzzy set P in Y.

III. JAYA ALGORITHM

Jaya algorithm (JA) is one of the as of late proposed population based algorithm for obtaining the solution of unconstrained and constrained optimization problems. This algorithm has a specific advantage of simplicity in its application to a problem and it is free from any kind of algorithm-specific parameters. It precisely involve the adjustment of common control parameters like number of iterations, population size, and design variables for better results [1], [2]. Although common control parameters are present in JA but the tedious work of adjusting algorithm-specific parameters for different applications is completely omitted. These advantages provide a great scope for application of JA to distinct engineering problems. The structure of this algorithm is fundamentally derived from the motility of obtained solution in the direction of best solution and far from worst solution. Therefore, a best variegation and escalation of the search process is reached. The algorithm constantly put efforts to acquire the best solution (i.e. nearer to success) simultaneously moves far from worst solution (i.e. attempt to obviate failure) [2]. In sanskrit, 'JAYA' means victory or triumph likewise This algorithm endeavor to become winning by achieving the best solution.

Let Z be objective function that to be minimized. Suppose there are T number of total populations $(p=1,2,...,T)$ and S number of design variables $(q=1,2,...,S)$ for every population. The first population is generated randomly in between their boundaries of size $(T\times S)$. If $y^r_{p,q}$ is the q^{th} design variable of the p^{th} population during r^{th} iteration then, $y^r_{p,q}$ is updated according to:

$$y^{r+1}_{p,q}=y^r_{p,q}+\alpha^r_q\left(y^r_{p,b}-|y^r_{p,q}|\right)-\beta^r_q\left(y^r_{p,w}-|y^r_{p,q}|\right) \qquad (3.1)$$

Where, $y^r_{p,b}$ and $y^r_{p,w}$ are the best and worst solution of the q^{th} design variable at the r^{th} iteration, respectively. $y^{r+1}_{p,q}$ represents improved value of the $y^r_{p,q}$. α^r_q and β^r_q are the random numbers of q^{th} design variable at the r^{th} iteration in the interval $[0,1]$. A greedy selection approach is taken to choose the best solution that taking part in the next iteration. It is expressed as:

$$Y^{r+1}_p=\begin{cases}y^{r+1}_p, & \text{if } Z\left(y^{r+1}_p\right)\le Z\left(y^r_p\right),\\ y^r_p, & \text{if } Z\left(y^{r+1}_p\right)\ge Z\left(y^r_p\right).\end{cases} \qquad (3.2)$$

Where, Y^{r+1}_p is the chosen solution that is involved in $(r+1)^{th}$ iteration. This process of improvement and selection of population is carried until the stopping criteria is achieved.

IV. BINARY JAYA ALGORITHM

Binary Jaya algorithm (BJA) is the binary edition of JA where the updated population $Y^r_{p,q}$ is represented in binary form. Each design variable of the first population y is randomly produced in the range $(0,1)$ and then, converted into binary form according to following rule:

$$y_{p,q}=\begin{cases}1, & \text{if } \operatorname{rand}(\ )\ge 0.5,\\ 0, & \text{otherwise.}\end{cases}$$

(4.1)

Where, $y_{p,q}$ is the binary form of the q^{th} design variable of the p^{th} population.

The value of $y_{p,q}$ obtained in (4.1) is updated according to (3.1).

The updated value $y_{p,q}$ is then transformed to a value in the range $(0,1)$ by the means of *tanh* transformation. It is expressed mathematically as:

$$\tanh\left(\left|Y_{p,q}\right|\right)=\frac{e^{\left(\left|2Y_{p,q}\right|\right)-1}}{e^{\left(\left|2Y_{p,q}\right|\right)+1}}. \qquad (4.2)$$

The transformed value of $Y_{p,q}$ obtained using equation (4.2) is then represented in binary form according to:

$$Y_{p,q}=\begin{cases}1, & \text{if } \operatorname{rand}(\ )<\tanh\left(\left|Y_{p,q}\right|\right),\\ 0, & \text{otherwise.}\end{cases} \qquad (4.3)$$

A greedy selection approach described by (3.2) is adopted to form new updated population taking part in the next iteration. This process is carried until the stopping criteria is achieved.

V. HYBRID JAYA ALGORITHM FOR SOLVING MULTI-OBJECTIVE 0-1 INTEGER PROGRAMMING PROBLEM

Jaya algorithm is a recently developed evolutionary approach which gives the approximate solution however the obtained solution isn't assured its existence as a feasible or globally optimum solution. Sometime, it might be the objective function that entrapped in local optima. So, to overcome this situation, we have improve the Jaya algorithm by bring in the conception of exponential membership function and binary numbers. Thus, the Jaya algorithm is improved while preserving its essential properties.

The Hybrid Jaya algorithm used for obtaining the solution of Multi-objective zero-one integer programming problem is shown as follows:

```
input: Parameters $(Z_1, Z_2, \cdots, Z_m, n)$
  output: Obtain the solution of Multi-objective 0-1 IPP.
    Solve $MO(0-1)IPP(Z_k, X)$.
      begin
      read : example
      while example=$MO(0-1)IPP$ do
        for $k=1$ to $m$ do
          enter matrix $Z_k$.
        end
        −| define the Multi-objective zero-one integer programming problem as stated by $\alpha$-level.
        −| determine best and worst solution for all objective functions.
        for $k=1$ to $m$ do
          $Z_k^{best}=\min(Z_k)$
          Subject to constraints of the Multi-objective zero-one IPP
        end
        for $k=1$ to $m$ do
          $Z_k^{worst}=\max(Z_k)$
          Subject to constraints of the Multi-objective zero-one IPP
        end
        −| specify exponential membership function for every objective function.
        for $k=1$ to $m$ do
          $\mu_{Z_k}^E(x)=\begin{cases}1, & \text{if } Z_k \le Z_k^{best}.\\ \frac{e^{-s\psi_k(x)}-e^{-s}}{1-e^{-s}}, & \text{if } Z_k^{best}<Z_k<Z_k^{worst}.\\ 0, & \text{if } Z_k \ge Z_k^{worst}, \forall k.\end{cases}$
          Where,
            $\psi_k(x)=\frac{Z_k-Z_k^{best}}{Z_k^{worst}-Z_k^{best}}, k=1,2,\cdots,m$ &
            $s$ is a non-zero parameter prescribed by the decision maker.
        end
        −| define SOOP model under given constraints from MOOP model.
        for $k=1$ to $m$ do
          $\max W=\prod_{k=1}^{m}\mu_{Z_k}$
          Subject to constraints of the Multi-objective zero-one IPP
        end
        −| find the solution of SOOP using Binary Jaya Algorithm.

        Procedure: BJA
        begin
          Generation=0;
            for $i=1$ to $n$ do
              $P(X)$ = Generate the initial population into binary form.
              $x_i=rand(0,1)$
              $X_i=\begin{cases}0 & \text{if } x_i<0.5,\\ 1 & \text{if } x_i\ge 0.5.\end{cases}$
            end
            for $(X\in P)$
              Evaluate $W(X)$.
              Evaluate $W_{best}$ and $W_{worst}$ solution from the population.
            end

        while (stopping criteria not met)
          Generation=Generation+1;
          for $i=1$ to $n$ do
            Update each old population by Jaya Algorithm.
              $X_i' = X_i+\alpha\left(X(W_{best})-|X_i|\right)$
                        $-\beta\left(X(W_{worst})-|X_i|\right).$
            Transform the updated value $X$ in the $(0,1)$ by using $\tanh$ transformation.
```

```
                $\tanh\left(\left|X_i^{'}\right|\right)=\frac{e^{\left(\left|2X_i^{'}\right|\right)-1}}{e^{\left(\left|2X_i^{'}\right|\right)+1}}.$
                Replace the transformed value of X in binary form.
                $X_i^{'}=\begin{cases}0 & \text{if } X_i^{'}<0.5,\\ 1 & \text{if } X_i^{'}\geq 0.5.\end{cases}$
                Compare the updated population and old population.
                Generate the new population by using greedy selection approach.
                $X_{i+1}=\begin{cases}X_i^{'}, & \text{if } W\left(X_i^{'}\right)\geq W\left(X_i\right),\\ X_i, & \text{if } W\left(X_i^{'}\right)\leq W\left(X_i\right).\end{cases}$
            end
            Repeat (Until stopping criteria not met).
        end(while)
    end( begin)
end.
```

To established the effectualness of our suggested algorithm, we have applied it to a real world project management problem. So, in the next section, we provide the mathematical formulation of the project management problem.

VI. PROJECT MANAGEMENT PROBLEM AND ITS SOLUTION BY HYBRIDE JAYA ALGORITHM

Suppose that project network $T=<Q,S,(t,c,r)>$ is compile of a finite set Q of junctions and a set $S\subset Q\times Q$ of paths with the crisp activities time, cost, and risk all are derived from the functions $t:S\rightarrow R^{+}\cup\{0\}$, $c:S\rightarrow R^{+}\cup\{0\}$, and $r:S\rightarrow R^{+}\cup\{0\}$ also affiliated to the paths, respectively. t_{mn}, c_{mn} and r_{mn} represent as the time, cost and risk of the activity $(m,n)\in S$, respectively. The component of flow that inject in the project at starting junction must quit at completing junction is the assumption that taken by all linear programming formulation. Let x_{mn} be the design variable representing the quantity of flow in $(m,n)\in S$, As we know that exclusively one component of flow should be in whatever path at a particular time, so the design variable x_{mn} should adopt the binary numbers (0 or 1) only. The construction of Project Network problem with p junctions is as shown in model I:

Model I:

$Z=\min[Z_1,Z_2,Z_3]$

$$=\min\left\{\sum_{m=1}^{p}\sum_{n=1}^{p}t_{mn}x_{mn};\sum_{m=1}^{p}\sum_{n=1}^{p}c_{mn}x_{mn};\sum_{m=1}^{p}\sum_{n=1}^{p}r_{mn}x_{mn};\right\}.$$

Subject to the constraints:

$$\sum_{n=1}^{p}x_{1n}=1;$$

$$\sum_{n=1}^{p}x_{mn}=\sum_{o=1}^{p}x_{om},m=2,\cdots p-1;$$

$$\sum_{o=1}^{p}x_{op}=1;$$

$$x_{mn}=0\text{ or }1,(m,n)\in A.$$

The goal is to minimize the taken time, cost and risk of the project network from junction 1 to junction p. The flow is preserved i.e. Flow can't be produced neither destroyed during the project network and for this we have flow conservation equations which is represent by the constraints. The critical path for this project network consists of a set of activities $(m,n)\in S$ from the initial to the end where every activity in the path corresponds to the optimum design variable $x_{mn}^{*}=1$ in the optimum solution of Model I. The taken time, cost and risk required to accomplish the project are derived as the minimal objective functions Z_1,Z_2 and Z_3 of Model I respectively.

Suppose the work is made up of labors with labelled as $R,S,\cdots,Y,Z$ and have the following relationships: Suppose $C<A,B$ means A and B can't start until C is accomplished; $A,B<C$ means C can't start until both A and B are accomplished. By using the above symbols, construct the problem having the following restrictions: $R<U,V;S,U<W;T<X;S<Y;W,X<Z.$

Find the optimal path of the work when the time, cost and risk required for each labor is given as follows:

Table I: Values of different criteria from single decision maker

Labors	R	S	T	U	V	W	X	Y	Z
Time	23	8	20	16	24	18	19	4	10
Cost	200	45	185	170	210	175	180	30	60
Risk	18	7	15	13	20	14	16	6	10

The above project network problem is formulated mathematically as shown in model (I) using the given data values of the problem in Table I.

Model II:

$Z=\min[Z_1,Z_2,Z_3]$

Where

$Z_1=20x_{12}+23x_{13}+8x_{14}+19x_{25}+16x_{34}+18x_{46}+4x_{57}+10x_{67}+24x_{37};$

$Z_2=185x_{12}+200x_{13}+45x_{14}+180x_{25}+170x_{34}+175x_{46}+30x_{57}+60x_{67}+210x_{37};$

$Z_3=15x_{12}+18x_{13}+7x_{14}+15x_{25}+13x_{34}+14x_{46}+6x_{57}+10x_{67}+20x_{37};$

Subject to the constraints:

$x_{12}+x_{14}+x_{13}=1;$
$x_{12}=x_{25};$
$x_{13}=x_{34}+x_{37};$
$x_{25}=x_{57};$
$x_{14}+x_{34}=x_{46};$
$x_{46}=x_{67};$
$x_{57}+x_{67}+x_{37}=1;$ where, $x_{ij}=0$ or $1, \forall i, j.$

Solving this problem with Hybrid Jaya algorithm by using the sequential steps of the algorithm as stated in section 5, we obtained the following optimal solution $Z_1^*=36, Z_2^*=280, Z_3^*=31$ and the optimal path is $1-4-6-7$.

VII. RESULTS AND COMPARISONS

The obtained results of fuzzy programming technique (by linear membership function) and Hybrid Jaya algorithm are equal with same optimal path and optimal values. As the fuzzy programming technique is an exact method so it has ability to generate optimum solution but to find an optimum solution of a complex problem is more complicated as compared to the evolutionary approaches. The computational time taken by Hybrid Jaya algorithm for obtaining the solution in MATLAB-2018 software is nearly 2 seconds. The results of both these methods are shown in the following table:

Table II: Comparison of Fuzzy programming technique and Hybrid Jaya algorithm

Fuzzy Programming Technique [15]	Hybrid Jaya Algorithm
$Z_1^*=36$	$Z_1^*=36$
$Z_2^*=280$	$Z_2^*=280$
$Z_3^*=31$	$Z_3^*=31$

The numerical example stated above was also solved using JA but it failed to achieve optimal solution of the problem. As JA is an evolutionary approach here it gives an approximate solution of the above stated problem but it failed to give even any feasible solution of the problem.

The suggested algorithm also has been verified for problems involving up to 300-variables whereas fuzzy programming technique becomes more tedious and difficult for solving such problems.

VIII. CONCLUSION

As we all know that classical approaches are failed to give multiple best solutions for MOOP in a single simulation run whereas evolutionary approaches give multiple best approximate solutions in a single simulation run but approximate solution may not be even feasible solution or may be solution trapped in local optima. Moreover, exact method gives assurance to produce an optimal solution but it is very difficult to handle complex problem. We have proposed a Hybrid Jaya algorithm with some modification in basic Jaya algorithm by introducing the concepts of binary numbers and exponential membership functions. The Hybrid Jaya algorithm successfully works for the complex Multi-objective 0-1 IPP. For the simplicity, here we have solved a real world Multi-objective 0-1 Integer programming problem with this suggested algorithm and the obtained results were compared with the results of fuzzy programming technique. We found that the suggested algorithm is evenly good or better for obtaining the solution of multi-objective 0-1 integer programming problem.

REFERENCES

1. R. V. Rao, V. J. Savani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems" in *Computer-Aided Design*, vol. 43, No. 3, Elsevier, 2011, pp. 303-315.
2. R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems" in *International Journal of Industrial Engineering Computations*, vol. 7, No. 1, 2016, pp. 19-34.
3. Kumar Abhishek, V. Rakesh Kumar, S. Datta, and S. S. Mahapatra, "Application of jaya algorithm for the optimization of machining performance characteristics during the turning of cfrp (epoxy) composites: comparison with tlbo, ga, and ica" in *Engineering with Computers*, vol. 33, No. 3, Springer, 2017, pp. 457-475.
4. W. Warid, H. Hizam, N. Mariun, and N. Abdul-Waheb, "Optimal power flow using the jaya algorithm" in *Energies*, vol. 9, No. 9, Multidisciplinary Digital Publishing Institute, 2016, pp. 678.
5. S. P. Singh, T. Prakash, V. P. Singh, and M. G. Babu, "Analytic hierarchy process based automatic generation control of multi-area interconnected power system using jaya algorithm" in *Engineering Application of Artificial Intelligence*, vol. 60, Elsevier, 2017, pp. 35-44.
6. S. Phulambrikar, "Solving combined economic emission dispatch solution using jaya optimization algorithm approach" in *International Research Journal of Engineering and Technology*, vol. 3, 2016, pp. 501-512.
7. R. V. Rao and G. G. Waghmare, "A new optimization algorithm for solving complex constrained design optimization problems" in *Engineering optimization*, vol. 49, No. 1, Taylor & Francis, 2017, pp. 60-83.
8. S. Mishra and P. Ray, "Power quality improvement using photo-voltaic fed dstatcom based on jaya optimization" in *IEEE Transactions on Sustainable Energy*, vol. 7, No. 4, IEEE, 2016, pp. 1672-1680.
9. R. V. Rao, D. P. Rai, J. Ramkumar, and J. Balic, "A new multi-objective jaya algorithm for optimization of modern machining processes" in *Advances in Production Engineering & Management*, vol. 11, No. 4, University of Maribor, Faculty of Mechanical Engineering, 2016, pp. 271.
10. R. V. Rao and D. P. Rai, "Optimization of selected casting processes using jaya algorithm" in *Materials Today: Proceedings*, vol. 4, No. 10, Elsevier, 2017, pp. 11056-11067.
11. T. Prakash, V. P. Singh, S. P. Singh and S. R. Mohanty, "Binary jaya algorithm based optimal placement of phasor measurement units for power system observability" in *Energy Conversion and Management*, vol. 140, 2017, pp. 34-35.
12. R. Azizipanah-Abarghooee, P. Dehghanian and V. Terzija, "Practical multi-area bi-objective environmental economic dispatch equipped with a hybrid gradient search method and improved jaya algorithm" in *IET Generation, Transmission & Distribution*, vol. 10, No. 14, IET, 2016, pp. 3580-3596.
13. R. V. Rao and A. Saroj, "Multi-objective design optimization of heat exchangers using elitist jaya algorithm" in *Energy Systems*, vol. 2, No. 9, 2016, pp. 305-341.
14. R. V. Rao and A. Saroj, "A self-adaptive multi-population based jaya algorithm for engineering optimization" in *Swarm and Evolutionary computation*, vol. 37, Elsevier, 2017, pp. 1-26.
15. R. Verma, M. P. Biswal, and A. Biswas, "Fuzzy programming technique to solve multi-objective transportation problems with some non-linear membership functions" in *Fuzzy sets and systems*, vol. 91, No. 1, Elsevier, 1997, pp. 37-43.

AUTHORS PROFILE

Surbhi Tilva: Currently a research scholar in Applied Mathematics and Humanities Department at S. V. National Institute of Technology, Surat. Completed Post graduation in applied mathematics from M. S. University, Vadodara with gold medal. Area of research work is an evolutionary approach: Jaya Algorithm. She qualified GATE-2017 and GATE-2018 with best GATE Score is 376, AIR-434. She also qualified NET-2017(Dec.) with CSIR, JRF-108 and NET-2018(June) with CSIR, JRF-63. She also qualified the GSET-2017.

Jayesh Dhodiya: He is Associate professor at SVNIT, Surat. He secure gold medal in his Post-graduation Program. He has been completed five Research and Development project. His area of research interest is optimization technique, mathematical modeling and simulation knowledge based system, advance operation research, and data mining. His work has been published in so many journals like Journal of Operation Research, Journal of Science, Management World and Journal of Management. He has membership of Society of Special Function, SRM, Indian Mathematical Society, and Gujrat Ganit Mandal. He organized many conferences/workshops/seminars and given an expert talk at various places.