Analyzing and Implementing Metaheuristic GWO and FireFly Algorithms aiming to develop a Fault-Tolerant Hybrid GWOFF Algorithm

RAJBHUPINDER KAUR[1]
DEEPAK KUMAR[2]
BAL KRISHAN[3]

[1]Research Scholar (Ph.D.)
Guru Kashi University
Bathinda, Punjab, India
er.rajbhupinder@gmail.com
[2]Associate Professor
Guru Kashi University
Bathinda, Punjab, India
Dr.D.K.Mehta81@gmail.com
[3]Assistant Professor
Yadavindra College of Engineering
Bathinda, Punjab, India
balkrishan_76@rediffmail.com

Abstract. The approach adopted to handle hard problems is known as metaheuristics. The problem is considered as hard if discovering the optimal solution for it may not be always possible within the stipulated time. Discovering a single solution to a problem is easy and can be accomplished extremely fast, but finding the best possible solution to the same problem is very long. Optimization algorithms are intended to bridge this gap. The research paper aims at solving the problems for finding the optimal solution for two popular metaheuristic algorithms, GWO (Grey Wolf Optimization) and FF (Firefly) algorithms. Both the metaheuristics algorithms are studied and implemented. The two technical features comprised of metaheuristic algorithms are exploration and exploitation. The optimal solution has been evaluated alongside Makespan and Utilization Rate for both GWO (Grey Wolf Optimization) and FF (Firefly) algorithms. The lower value of the Makespan and higher Utilization Rate is always desirable. Both the algorithms have been modified via using mathematical functions to enhance the readings concerning performance evaluation parameters. The GWO is been modified via developing a hybrid version comprising GWO and PSO (Particle Swarm Optimization) algorithms denoted as the Hybrid Modified GWOPSO algorithm. The Firefly algorithm too has been modified and is denoted as a Modified FF algorithm. The conducted modifications have been measured via different performance evaluation parameters. Finally, the fault tolerance factor is considered and the modified versions like Modified GWOPSO and Modified FF are hybridized to develop a new proposed algorithm Hybrid GWOFF (Hybrid Grey Wolf optimizer Firefly) algorithm and its performance have been evaluated with and without fault tolerance by using different performance parameters.

Keywords: Fault Tolerance, Firefly, Grey Wolf Optimization, Optimal solution.

(Received May 1st, 2021 / Accepted May 21st, 2021)

1 Introduction

Algorithms with stochastic constituents were frequently denoted as a heuristic in the past, however, the modern literature tends to denote them as metaheuristics. The usage of a metaheuristic for a problematic scenario is acceptable bearing in mind the practical time in which it can discover a possible solution [1, 21]. It might not assure the best solution, but it can guarantee the nearly optimal solution, because, in some problems, it is impossible to examine for every combination in a reasonable time [15, 3]. The two technical features comprised of metaheuristic algorithms are exploration and exploitation. These two features are often known as diversification and intensification respectively. Exploration is concerned with exploring the global space, producing varied solutions at a global scale; and exploitation involves the usage of local and global information to create the best solution [18, 6]. The equilibrium between these two features is important. If exploitation takes the lead, it can quicken the process and respond with a local optimum. However, if exploration is the most used, the probability of finding the global optimum increases, but this slows down convergence [14, 13]. The optimal solution has been evaluated alongside Makespan and Utilization Rate for both GWO and FF algorithms. Both the algorithms have been modified via using mathematical functions to enhance the readings concerning performance evaluation parameters. Finally, these two algorithms are then hybridized to develop a new hybrid algorithm Hybrid GWOFF (Hybrid Grey Wolf Firefly) algorithm with and without fault tolerance. The two metaheuristics algorithms, GWO (Grey Wolf Optimization) and FF (Firefly) algorithms are studied and implemented.

1.1 Grey Wolf Optimization

Several meta-heuristic algorithms dedicated to deal with optimization complications came into existence in recent years and Grey wolf optimization is one such algorithm. Grey wolf algorithm enjoys resilient optimal research competence. The working principle behind the GWO algorithm is analogous to the hunting mechanism and leadership pyramid of wolves to explore the targets [11, 8]. Grey wolves prefer remaining in groups with each group comprising 5 to 12 wolves on average. The four levels of wolves? hierarchy are mentioned below [7, 17].

- Alpha (α) - This is the first level hierarchy. Its job is to make decisions related to hunting, find a place to sleep, and plan walk time, etc.
- Beta (β) - The job of beta wolves is to assist alpha candidates in making decisions or engaging in supplementary actions.
- Delta (δ) - Delta wolves work under the expertise of alpha and beta wolves. These are accountable for inspection, patrol, lookout, and other responsibilities.
- Omega (ω) - Omega wolves have to defer to the alpha, beta, and delta wolves. The task of omega wolves is to preserve the integrity of the hierarchical structure.

The grey wolves are smart enough to surround the prey with the capability to pinpoint its location. Alpha heads the complete hunting process [5]. Often in complex situations, locating the prey at the beginning becomes cumbersome. The first three solutions of alpha, beta, and delta hold detailed information related to the position of the prey. The updation in the position of other wolves depends on positions of alpha, beta, and delta [26, 9]. The hunting process adopted by a grey wolf is shown in Figure 1.

The prey is encompassed by the grey wolves and hunting is carried out. The mathematical equations for this behavior are mentioned below.

E = |N.W_p(c) - W (c)|

W (c+1) = W_p(c) - M.E

where

E - Denotes to the distance amid a wolf and the prey

W - Denotes to grey wolf?s position vector

W_p - Denotes to prey?s position vector

c - Denotes to the in-progress iteration

M and N - Denotes to the coefficient vectors which are calculated as

M = 2b.v_1 - b

N = 2v_2

where b diminishes linearly along with the number of iterations from 2 to 0. v_1 and v_2 are the random vectors in [0,1].

1.2 Firefly Algorithm

The firefly algorithm is grounded on the flashing patterns and behavior of fireflies. In essence, FA uses the following three idealized rules [23, 22]:

- Being unisexual, the fireflies are attracted to other fireflies irrespective of their sex.
- The intensity of attractiveness depends on the brightness of the fireflies which diminishes with the increase in distance between the fireflies. As per

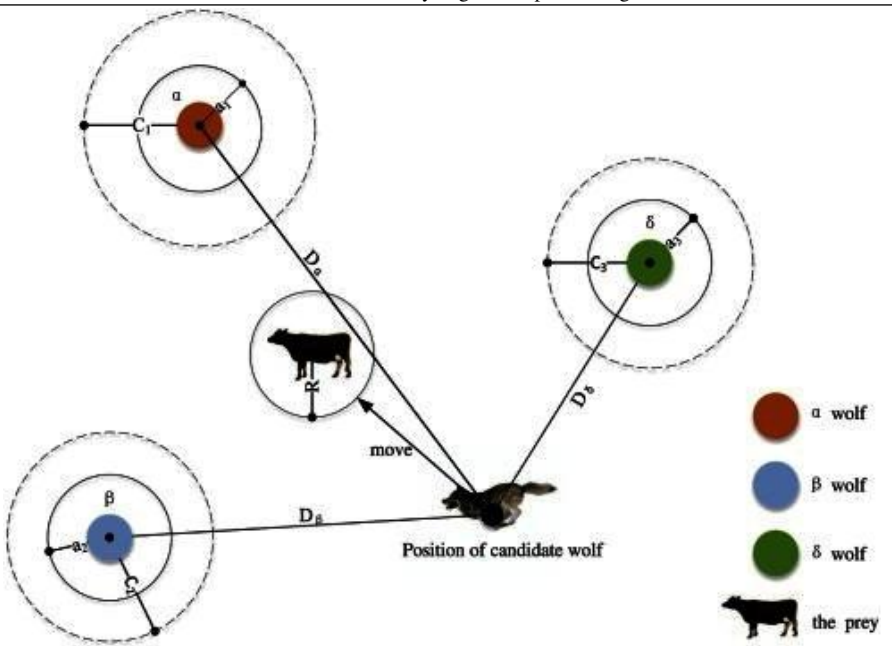

Figure 1: Hunting Process Adopted by Grey Wolves [5]

principle, the lesser brighter fireflies are attracted towards the brighter ones [21, 24].

- In the absence of the brighter firefly, the fireflies would adopt a random path. The brightness of a firefly is determined by the landscape of the objective function [4, 16].

Firefly algorithms have two major advantages when compared with other algorithms. The principle behind the working of the Firefly algorithm is the intensity of attraction which decreases with the increase in the distance. The total population can routinely be divided into subgroups and individual groups can swarm about each mode or local optimum. The global best solution can be searched among these modes. The second advantage is that this division permits the fireflies the capability to find the optima simultaneously if the size of the population is considerably higher as compared to the number of modes. The scenario for the operation of the Firefly algorithm is shown in Figure 2.

The movement of a firefly i is attracted to another, more attractive (brighter) firefly j is determined by

x_{it} + 1 =(x_{it} + β_0e)-γrij2(x_{jt} - x_{it}) + $\alpha\varepsilon_{it}$

where β_0 is the attractiveness at the distance r = 0, γ is a scale-dependent parameter controlling the visibility of the fireflies and α is a scaling factor controlling the step sizes of the random walks. The second term is due to the attraction. The third term is randomization with α being the randomization parameter, and ε_{it} is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. If β_0=0, it becomes a simple random walk.

2 State of Art

Jui-Sheng Chou1&Ngoc-Tri Ngo, 2017 [2] proposed an enhanced metaheuristic, nature-inspired algorithm termed as MFA (Modified Firefly algorithm). MFA comprises different metaheuristic components like Gauss / mouse chaotic maps, Levy flight, and adaptive inertia weight along with a conventional FA (Firefly algorithm) to enhance the capability for optimizing. The purpose of Gauss/mouse maps is to fine-tune the attractiveness parameter of FA. The adaptive inertia weight is intended to control the local exploitation and global exploration of the search process under consideration. The purpose of Levy flight is to figure out the exploitation of the MFA. The projected MFA was assessed by associating its enactment in resolving a series of benchmark functions with those of the FA and other well-known optimization algorithms. The effectiveness of the MFA has been established via its solutions to the three multi-dimensional structural design optimization problems.

Among the considered algorithms, MFA delivered the best results. Investigational consequences exposed that the anticipated MFA is more competent and active

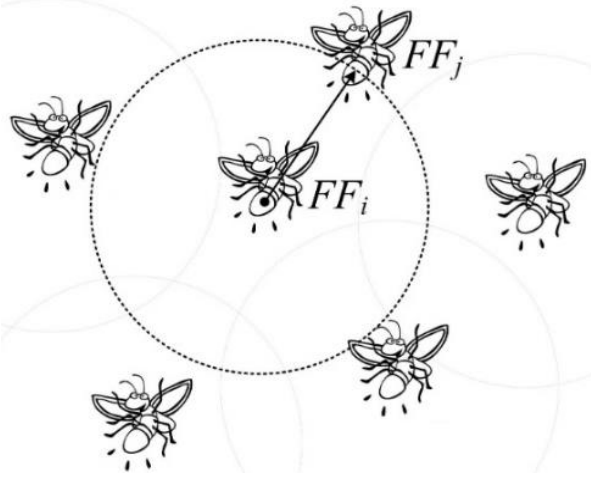

Figure 2: The Operational Scenario of Firefly Algorithm

than the compared algorithms. Narinder Singh and S. B. Singh, 2017 [19]; proposed a new hybrid nature-inspired algorithm termed as HPSOGWO formed via hybridization of PSO (Particle Swarm Optimization) and GWO (Grey Wolf Optimization). The primary emphasis laid by the authors is to enhance the capability of exploitation in PSO alongside enhancing the ability of exploration in GWO. The authors made use of a few unimodal, multimodal and fixed dimension multimodal test functions to check the performance and solution quality of HPSOGWO. The obtained results indicate the effectiveness of the hybrid HPSOGWO against the PSO and GWO variants in accordance with solution stability, solution quality, the capability to find the global optimum, and convergence speed. Mehak Kohli, Sankalap Arora, 2018 [10]; proposed a chaos theory into the GWO (Grey Wolf Optimization) algorithm aiming to speed up its global convergence speed. Initially, thorough studies are conducted on thirteen standard constrained benchmark problems with ten diverse chaotic maps to discover out the most competent one. Thereafter, the chaotic GWO is equated with the conventional GWO and few other popular meta-heuristics algorithms like FA (Firefly Algorithm), FPA (Flower Pollination Algorithm) and PSO (Particle Swarm Optimization) algorithm. The performance of the CGWO algorithm is also authenticated utilizing five controlled engineering design glitches. The obtained results exhibited that with a suitable chaotic map, CGWO can outperform standard GWO, with very decent performance in contrast with supplementary algorithms and application to controlled optimization problems. Fazli Wahid, Rozaida Ghazali, 2019 [20]; stated the features of the FA (Firefly) algorithm as nature inspired, meta heuristic, and stochastic algorithm intended to solve various optimization problems.

The authors recommended FA as easy to implement an algorithm. The FA algorithm operates in three stages: initialization, changing positions, and the termination stage. The major drawback encountered in the working of FA is witnessed in its final stage where after a fixed number of iterations, the improvement in the desired solution becomes static. In this paper, this matter is fixed by familiarizing pattern search (PS) at the finishing stage of standard FA when no more progress is witnessed in the solution quality. The anticipated method comprises three stages. In the primary stage, the parameters of standard FA are initialized. In the firefly shifting position step, the randomization factor is used to describe the solution in the individual iteration of operational phases. In the last stage, the augmented values gained from the FA throughout its maximum number of iterations are set as inputs to the pattern search algorithm. The pattern search enhances the values gained in the maximum iterations of standard FA. The anticipated method has been termed as FA-PS in which PS has been utilized to acquaint with enhancement in the solution quality of standard FA. The established method has been smeared to numerous types of maximization and minimization functions and the performance has been linked with standard FA and genetic algorithm in terms of accomplishing the greatest optimal values for the functions being considered. A noteworthy development has been witnessed in the solution quality of FA. Zhihang Yue, Sen Zhang, and Wendong Xiao, 2020 [25]; stated that GWO (Grey Wolf Optimization) falls under the category of the meta-heuristic algorithm with

strong optimal search capability. However, GWO often converges to the local optimum. Alongside GWO, the author elaborated on FWA (Fireworks algorithm). The authors hybridized the two algorithms to attain the global optima successfully. The suggested algorithm is a mix of exploration capability of FWA and exploitation capability of GWO. The authors set a balance coefficient. The sixteen benchmark functions have been used in the conducted research. By altering the balance coefficient, the FWGWO algorithm can circumvent the local optimal value as much as possible and has a higher convergence speed. The comparison of the developed hybrid FWGWO algorithm is compared with nine other algorithms comprising the conventional GWO, enhanced GWO, conventional FWA, and augmented GWO. The achieved results indicate the FWGWO is quite effective in improving the convergence speed and global search capability of the FWA and GWO. Yuanyuan Liu, Jiahui Sun, Haiye Yu, Yueyong Wang, and Xiaokang Zhou, 2020 [12]; proposed an improved GWO algorithm based on DE (differential evolution) and OTSU algorithms and named it DE-OTSU-GWO. GWO algorithm is attached with Tsallis entropy, multi-threshold OTSU, and DE algorithm. The updation of the population is done using DE and GWO algorithm via Tsallis entropy considering crossover steps. Tsallis entropy is intended to calculate the fitness quickly. The multi-threshold OTSU computes the fitness in the initial population and ensures the stability of the initial stage. CEC2005 benchmark function is used to test the performance of DE-OTSU-GWO. Equated with prevailing PSO (Particle Swarm Optimization) and GWO algorithms, the investigational outcomes exhibited that the DE-OTSU-GWO algorithm is more steady and precise in solving functions. Besides, associated with supplementary algorithms, a convergence behavior investigation demonstrated the high quality of the DE-OTSU-GWO algorithm. The OTSU algorithm progresses the precision of the overall algorithm while enhancing the running time. After an accumulation of the DE algorithm, the time complexity will upsurge, but the solution time can be reduced. Compared with PSO, GWO, DE-GWO, and 2D-OTSU-FA, the DE-OTSU-GWO algorithm has improved results in segmentation valuation.

3 Contribution And Implementation

This section comprises the detailed implementation of GWO, Hybrid GWOPSO (Grey Wolf Optimization Particle Swarm Optimization), Firefly, and Modified Firefly algorithms accompanied by flowcharts and algorithms. The obtained results have been analyzed by readings of different performance evaluation parameters like Makespan, Utilization Rate, Throughput, Waiting Time, and Turnaround Time.

3.1 Grey Wolf Optimization

Flowchart of GWO Figure 3 depicts the flowchart illustrating the working of the GWO algorithm.

Algorithm of GWO The algorithm for GWO is mentioned below.

- Initialize the wolf population W_i (i = 1, 2, ... , n)
- Initialize b, M, and N
- Calculate the fitness of three search agents

  Wα (Best search agent)

  Wβ (Second-best search agent)

  Wδ (Third-best search agent)
- while (c < Max number of iterations) for each search agent

  Apprise the position of the current search agent

  end for

  Update b, M, and N

  Calculate the fitness of all search agents Update Wα, Wβ, and Wδ

  c = c + 1

  end while

  return Wα

Implementation - GWO (Grey Wolf Optimization)

The different parameters considered with assigned values to execute the working of GWO are given in case 1.

Case 1

Number of search agents (Searchagents_no) = 30

Test functions that ranges from F1 to F23 (Function_name) = 'F1'

Maximum Number of iterations (Max_iteration) = 500

The readings obtained after the execution of GWO as per values assigned to the parameters are mentioned in Table 1, which shows the readings of the first 15 and last 15 iterations.

Results:

Makespan: 1.2160

Utilization Rate: 0.8224

Throughput = 0.231489362

Turnaround time in Deci seconds = 115.744681

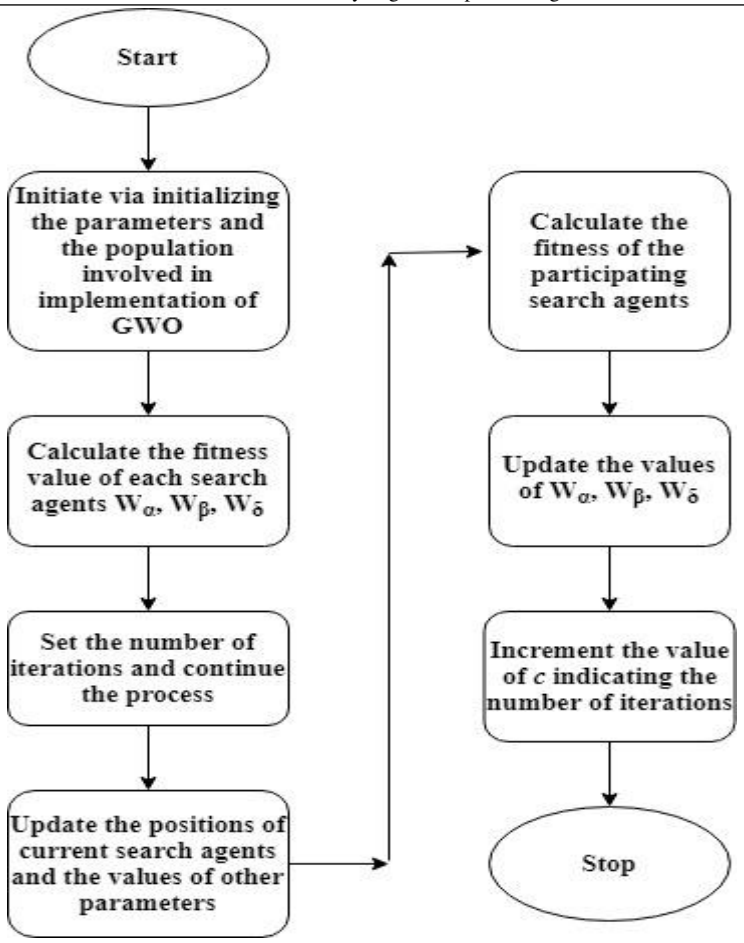

Figure 3: Working of GWO algorithm

Waiting time in Deci seconds = 115.288830

Burst time in Deci seconds = 0.455851

Optimum value of the objective function evaluated by GWO is: 7.9879

Figure 4 shows the graphical representation of the obtained results from the implementation of GWO as per defined readings. The figure on the left side shows the results in relevance with the parameter space and the figure on the right side shows the graph depicting the Best Score (Y-axis) against the number of iterations (X-Axis).

Modified Hybrid GWOPSO

Flowchart for Modified Hybrid GWOPSO

The proposed flowchart for Hybrid GWOPSO is shown in Figure 5.

Algorithm for Hybrid Modified GWOPSO

- Consider appropriate search agents (SearchAgents_no)
- Choose from the range of test functions from F1 to F23(Function_name)
- Fix maximum iterations (Max_iter)
- Load details of the selected benchmark function
- Initialize positions of alpha, beta, and delta wolves

  A_pos=zeros(1,dim); A_score=inf;

  B_pos=zeros(1,dim); B_score=inf;

  D_pos=zeros(1,dim);D_score=inf;
- Initialize the positions of search agents

  If the boundaries of all variables are equal and user enter a single number for both ub (upper bound) and lb (lower bound)

  if Boundary_no==1

  Positions=rand(SearchAgents_no,dim).*(ub-lb)+lb;

  End

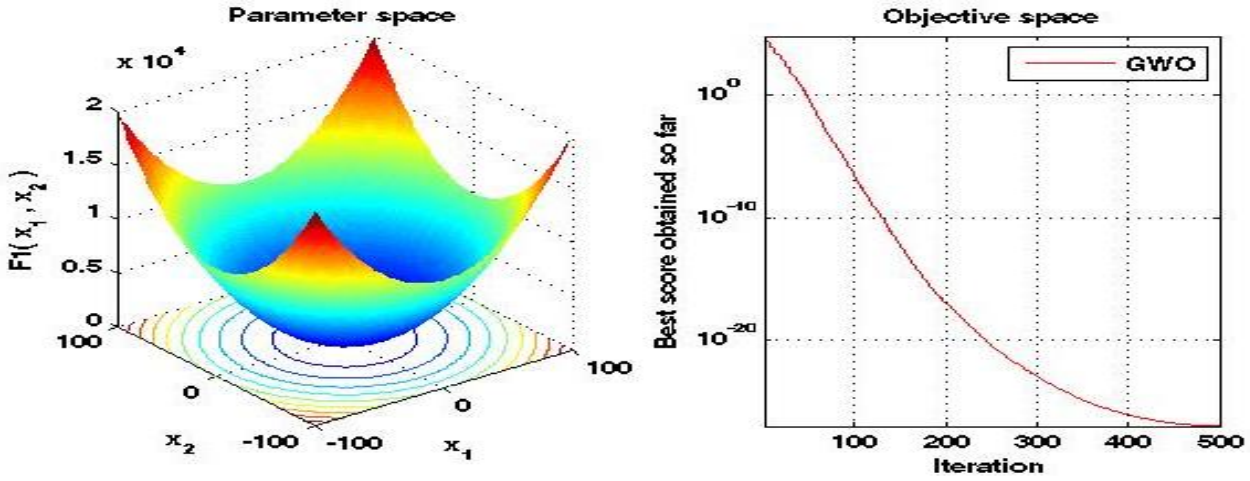

Figure 4: Results Obtained after Execution of GWO in Graphical Form

Table 1: Readings of performance evaluation parameters obtained after execution of GWO as per Case 1

Iteration No.	Burst Time	Waiting Time	Turnaround Time
1	0.003032	0	0.003032
2	0.006386	0.003032	0.009418
3	0.007086	0.009418	0.016504
4	0.008116	0.016504	0.02462
5	0.008871	0.02462	0.033491
6	0.009729	0.033491	0.043221
7	0.010855	0.043221	0.054076
8	0.011553	0.054076	0.065629
9	0.012684	0.065629	0.078313
10	0.013463	0.078313	0.091776
11	0.014189	0.091776	0.105965
12	0.014954	0.105965	0.120919
13	0.015608	0.120919	0.136527
14	0.016297	0.136527	0.152824
15	0.017187	0.152824	0.170011
CONTINUED			
486	0.443723	108.54499	108.988712
487	0.443723	108.54499	108.988712
488	0.444585	108.98871	109.433298
489	0.445442	109.4333	109.87874
490	0.446271	109.87874	110.32501
491	0.446977	110.32501	110.771987
492	0.449061	111.22032	111.669379
493	0.449773	111.66938	112.119152
494	0.450428	112.11915	112.56958
495	0.452021	113.02074	113.472763
496	0.452715	113.47276	113.925478
497	0.453728	113.92548	114.379206
498	0.454452	114.37921	114.833658
499	0.455172	114.83366	115.28883
500	0.455851	115.28883	115.744681

Else If each variable has a different lb and ub

if Boundary_no>1

for i=1:dim

ub_i=ub(i);

lb_i=lb(i);

Positions(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;

End

End

End

- Execute until the maximum number of iterations is reached.

  Calculate the objective function for each search agent.

  Apprise Alpha, Beta, and Delta.

  For Alpha

  if fitness<A_score (value of fitness is less than A_score)

  A_score=fitness; (assign fitness to A_score)

  A_pos=Positions(i,:); (update positions)

  end

  For Beta

  if fitness>A_score && fitness<B_score (value of fitness is more than A_score and less than B_score)

  B_score=fitness; (assign fitness to B_score)

  B_pos=Positions(i,:); (update positions)

  end

  For Delta

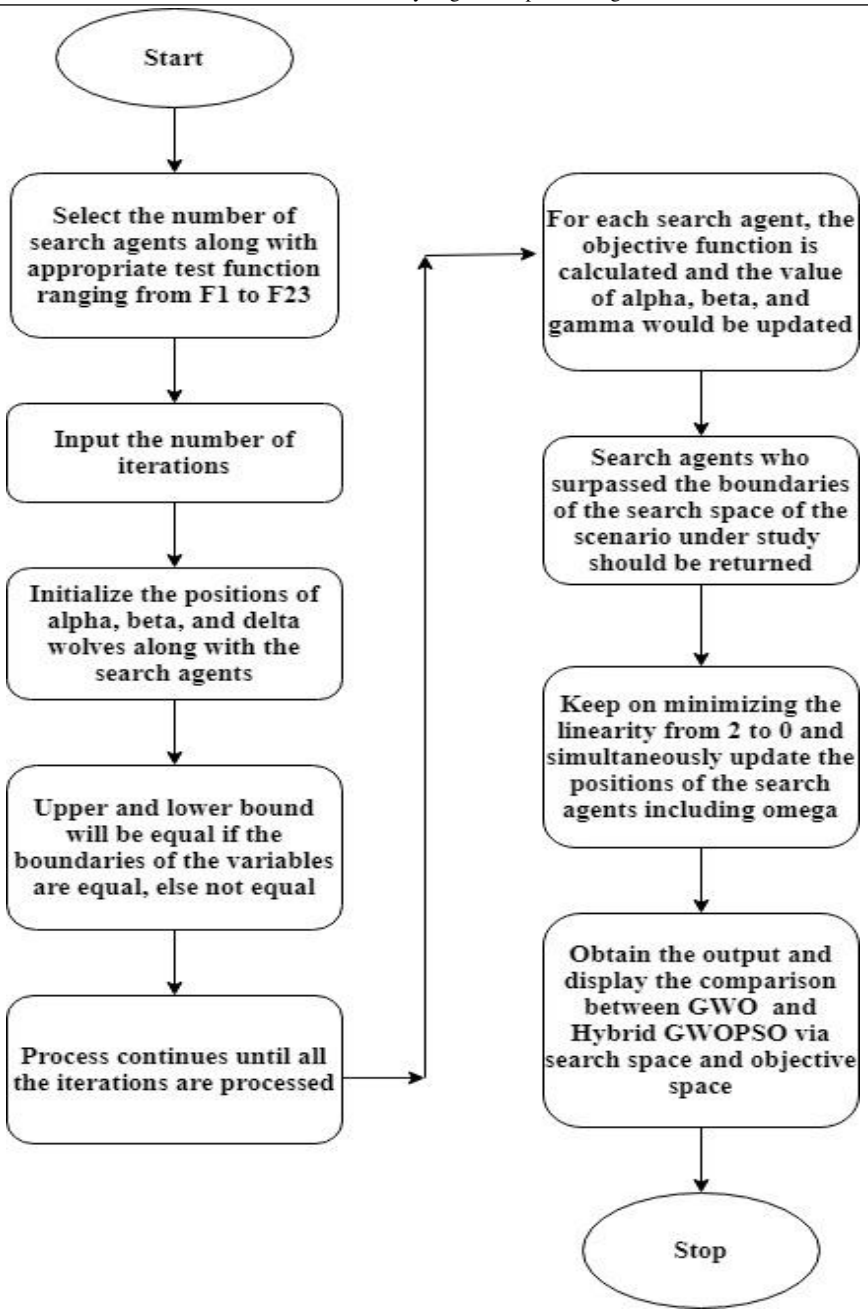

Figure 5: Flowchart of Proposed Hybrid GWOPSO Algorithm

if fitness>A_score && fitness > B_score && fitness < D_score

(value of fitness is more than A_score and B_score and less than D_score)

D_score=fitness; (assign fitness to D_score)

D_pos=Positions(i,:); (update positions)

End

- Return the search agents that go beyond the boundaries of the search space
- Calculate objective function for each search agent
- Update Alpha, Beta, and Delta

- Decrease linearly from 2 to 0.
  a=2-l*((2)/Max_iter)
- Update the Position of search agents including omegas
- Draw search space
- Draw objective space
- Obtain results

Implementation of Hybrid Modified GWOPSO

The different parameters considered with assigned values to execute the working of Modified Hybrid GWOPSO are given in case 2.

Case 2

Number of search agents (SearchAgents_no) = 30

Test function ranges from F1 to F23 (Function_name) = 'F1'

Maximum Number of Iterations (Max_iteration) = 500

The readings obtained after the execution of Modified Hybrid GWOPSO as per Case 2 are mentioned in Table 2,which shows the readings of the first 15 and last 15 iterations.

Results:

Makespan: 1.0460

Utilization Rate: 0.9560

Throughput = 0.3815871

Turnaround time in Deci seconds = 190.793550

Waiting time in Deci seconds = 190.045185

Burst time in Deci seconds = 0.748364

Optimal value of the objective function calculated by PSOGWO is: 708.5633

Figure 6 shows the graphical representation of the obtained results from the implementation of Modified Hybrid GWOPSO as per Case 2. The figure on the left side shows the results in relevance with the parameter space and the figure on the right side shows the graph depicting Best Score (Y-axis) against the number of iterations (X-Axis) in the case of GWO (blue line) and Hybrid Modified GWOPSO (red line). The graph depicts the Modified Hybrid GWOPSO algorithm scores over the GWO algorithm in terms of Best Cost achieved.

The comparative evaluation of considered evaluation parameters is shown in Table 3.

The readings for different parameters for GWO and Hybrid GWOPSO have been obtained in Table III. The Makespan for GWO has been recorded as 1.2160 and for the Modified Hybrid GWOPSO has been recorded as 01.0460. The lower is the value of Makespan, the better is the performance of the algorithm. The Utilization Rate is inversely proportional to the Makespan

Table 2: Readings of performance evaluation parameters obtained after execution of Modified Hybrid GWOPSO as per Case 2

Iteration No.	Burst Time (in seconds)	Waiting Time (in seconds)	Turnaround Time (in seconds)
1	0.003869	0	0.003869
2	0.008903	0.003869	0.012772
3	0.010308	0.02308	0.012772
4	0.012094	0.02308	0.035174
5	0.013515	0.035174	0.048739
6	0.015383	0.048739	0.064122
7	0.016882	0.064122	0.081004
8	0.018296	0.081004	0.0993
9	0.019702	0.0993	0.119002
10	0.021081	0.119002	0.140083
11	0.022605	0.140083	0.162689
12	0.024514	0.162689	0.187202
13	0.025909	0.187202	0.213111
14	0.027375	0.213111	0.240486
15	0.028826	0.240486	0.269312
CONTINUED			
486	0.7258	179.73721	180.46301
487	0.7273	180.46301	181.19031
488	0.728961	181.19031	181.91927
489	0.730483	181.91927	182.64975
490	0.732262	182.64975	183.38202
491	0.73384	183.38202	184.11586
492	0.73535	184.11586	184.85121
493	0.737406	184.85121	185.58861
494	0.738928	185.58861	186.32754
495	0.74043	186.32754	187.06797
496	0.74206	187.06797	187.81003
497	0.74358	187.81003	188.55361
498	0.745078	188.55361	189.29869
499	0.746497	189.29869	190.04519
500	0.748364	190.04519	190.79355

Table 3: The values of Performance Evaluation Parameters of GWO and Modified Hybrid GWOPSO algorithms

Parameters / Algorithms	GWO	Modified Hybrid GWOPSO
Makespan	1.216	1.046
Utilization Rate	0.8224	0.956
Waiting Time (deci seconds)	115.28883	190.045185
Turnaround Time (deci seconds)	115.744681	190.79355
Throughput	0.231489362	0.3815871
Optimal Value	7.9879	708.5633

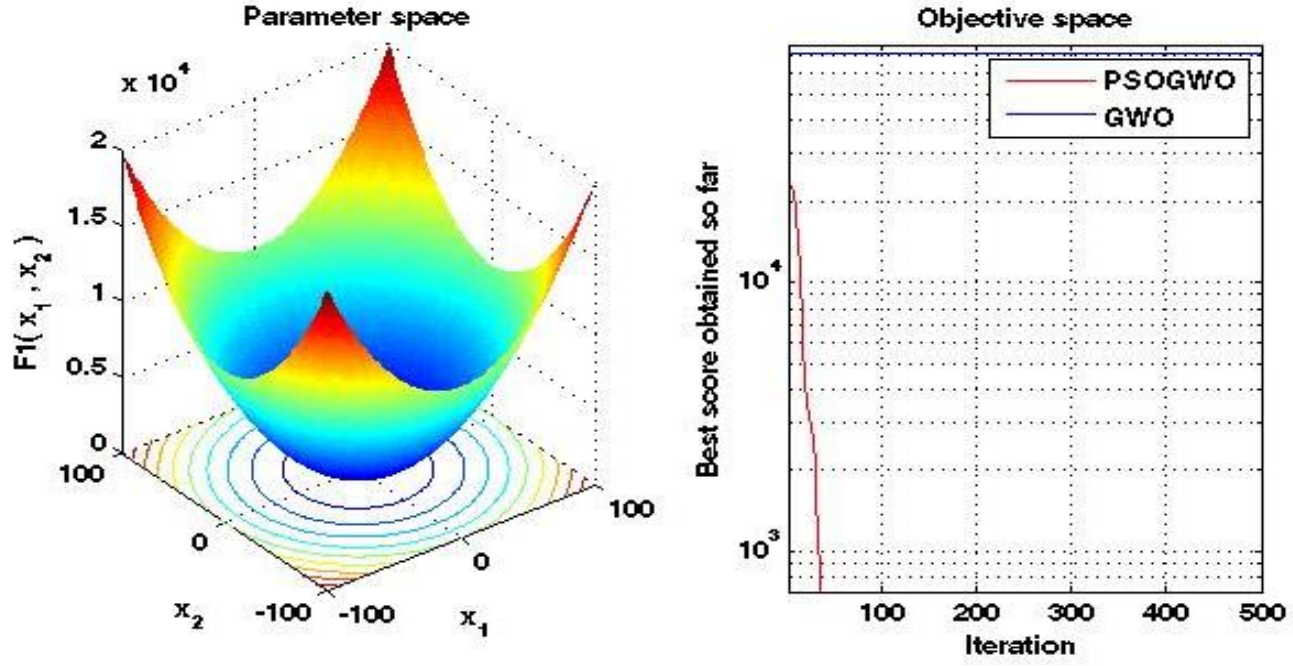

Figure 6: The Comparative Results of GWO and Hybrid GWOPSO as per readings of Case 2

and therefore Modified Hybrid GWOPSO has a higher Utilization Rate of 0.9560. The higher value of the Utilization Rate indicates the optimized use of available resources. Similarly, the Throughput for Modified Hybrid GWOPSO is 0.3815871 which is greater than GWO having a Throughput of 0.231489362. The Throughput is considered as the primary parameter to evaluate the performance of an algorithm. The Turnaround Time for Modified Hybrid GWOPSO is 190.793550 Deci seconds and for GWO is 115.744681 Deci seconds. The Waiting times of 115.288830 and 190.045185 Deci seconds have been recorded for GWO and Modified Hybrid GWOPSO respectively. The Optimal Value for GWO has been recorded to be 7.9879 and that for Modified Hybrid GWOPSO has been 708.5633. The greater value obtained for Optimal Value for Modified Hybrid GWOPSO indicates the superiority of Modified Hybrid GWOPSO over GWO.

Different fragments of Figure 7 illustrate the comparative graph of the performance evaluation parameters for GWO and Modified Hybrid GWOPSO. In all the fragments, the X-axis denotes the name of the algorithms under consideration. The Y-axis represents the readings of different performance evaluation parameters. In Fig. 7 (a), the Y-axis represents the reading of makespan, in Fig. 7 (b), Y-axis represents the utilization rate, in Fig. 7 (c), Y-axis denotes the throughput, in Fig. 7(d), Y-axis represents denotes the turnaround time in deci seconds, in Fig. 7(e), Y-axis represents the waiting time in deci seconds, and Fig. 7(f) represents the readings for optimal value.

So, from the discussion, it can be concluded that the Modified Hybrid GWOPSO scores over GWO in terms of Makespan, Utilization Rate, Throughput, Turnaround Time, Waiting Time, and Optimal Value.

3.2 Firefly Algorithm

This sub-section elaborates the implementation of the Firefly and Modified Firefly algorithm via means of flowcharts, algorithms, and implementations. Figure 8 shows the detailed flowchart of the Firefly algorithm.

Algorithm for Firefly algorithm

- Initialize the parameters (Number of Decision Variables, Decision Variables Matrix Size, Decision Variables Lower and upper bound, Maximum Number of Iterations, Number of Fireflies (Swarm Size), Light Absorption Coefficient, Attraction Coefficient Base Value, Mutation Coefficient, Mutation Coefficient Damping Ratio, and Uniform Mutation Range).
- Empty Firefly Structure

  firefly.Position = []

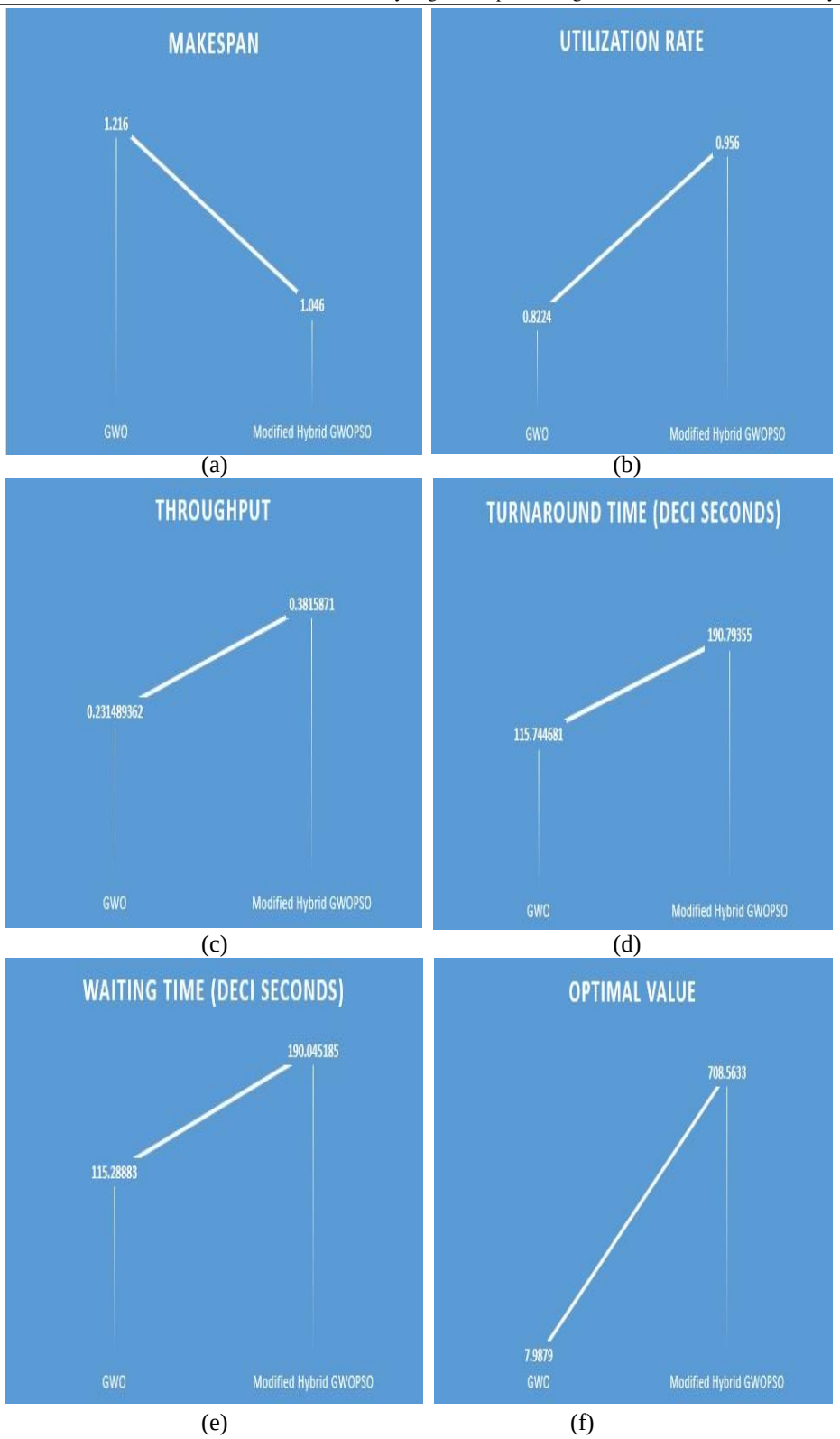

Figure 7: (a) Makespan, (b) Utilization Rate, (c) Throughput, (d) Turnaround Time, (e) Waiting Time, (f) Optimal Value

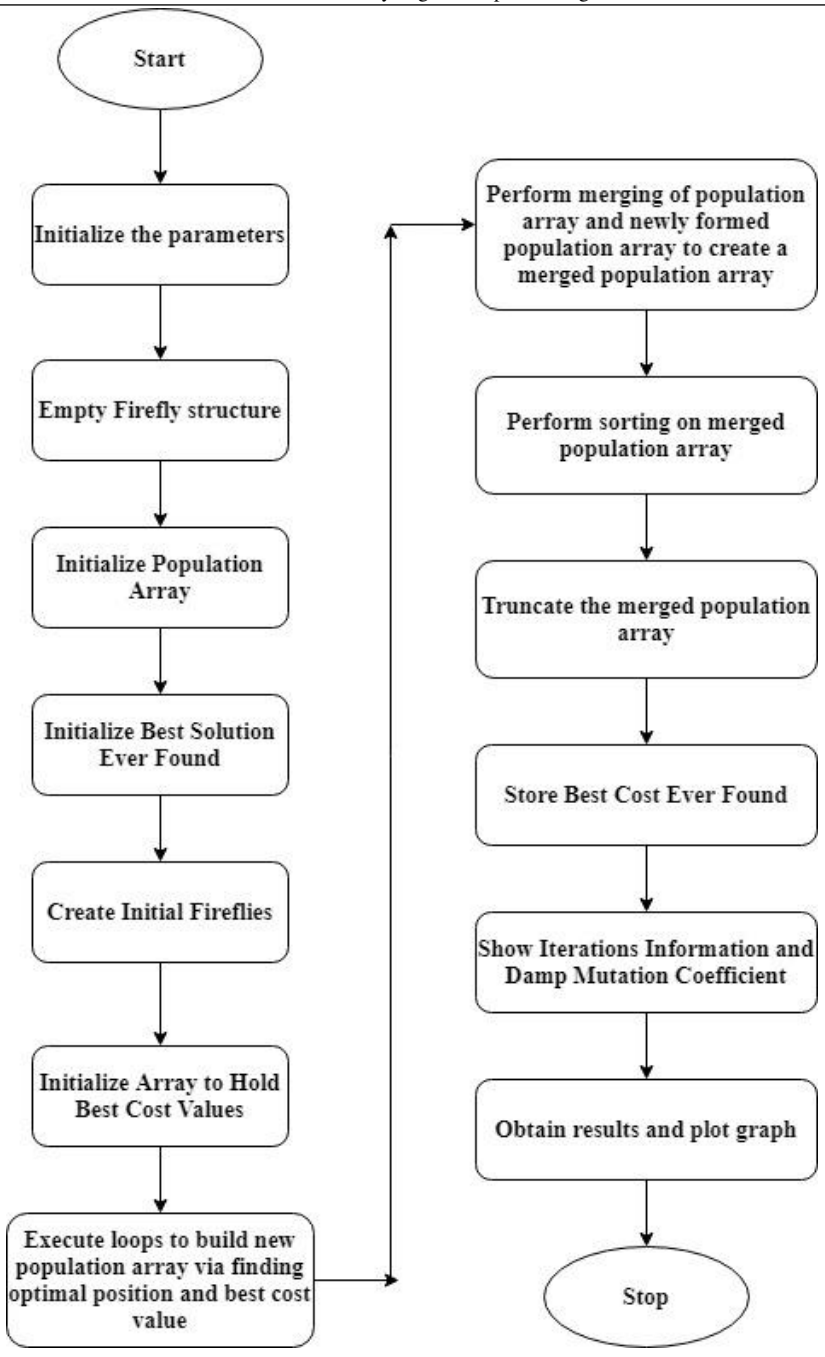

Figure 8: Flowchart of Firefly Algorithm

firefly.Cost = []

- Initialize Population Array

  pop = repmat(firefly, nPop, 1)

- Initialize Best Solution Ever Found

  BestSol.Cost = inf

- Create Initial Fireflies

  for i = 1:nPop

  pop(i).Position = unifrnd(VarMin, VarMax, VarSize);

  pop(i).Cost = CostFunction(pop(i).Position);

if pop(i).Cost <= BestSol.Cost

BestSol = pop(i);

end

end

- Initialize Array to Hold Best Cost Values

  BestCost = zeros(MaxIt, 1);

- Execute loops to build new population array newpop via finding an optimal position and best cost value.

- Perform merging of population array pop and newly formed population array newpop to form merged array pop.

  pop = [pop,newpop]

- Perform sorting on merged population array.

  [, SortOrder] = sort([pop.Cost])

  pop = pop(SortOrder)

- Truncate the merged population array.

  pop = pop(1:nPop)

- Store Best Cost Ever Found

  BestCost(it) = BestSol.Cost

- Show Iterations information

- Damp Mutation Coefficient

  alpha = alpha*alpha_damp

- Obtain results and plot graph.

Execution Parameters

The different parameters playing part in the execution of the Firefly algorithm are stated below with a brief description.

Number of Decision Variables - nVar
Decision Variables Matrix Size - VarSize = [1 nVar]
Decision Variables Lower Bound - VarMin
Decision Variables Upper Bound - VarMax
Maximum Number of Iterations - MaxIt
Number of Fireflies (Swarm Size) - nPop
Light Absorption Coefficient - gamma
Attraction Coefficient Base Value - beta0 Mutation Coefficient - alpha Mutation Coefficient Damping Rati - alpha_damp Uniform Mutation Range -delta = 0.05 *(VarMax - arMin)

Case 3:

The values assigned to the participating parameters are mentioned as :-

nVar = 5
VarSize = [1 nVar]
VarMin = -10
VarMax = 10
MaxIt = 300
nPop = 25
gamma = 1
beta0 =2
alpha = 0.2
alpha_damp = 0.98
delta = 0.05 * (VarMax ? VarMin)

Table 4: Optimal Solution for iterations obtained as per specifications of Case 3

Iteration No.	Optimal Solution
1	10377.216
2	733.6742
3	49.489
4	49.489
5	10.7325
6	4.9077
7	2.894
8	2.894
9	2.0964
10	2.0964
CONTINUED	
291	0.0001526
292	0.0001526
293	0.0001379
294	0.0001379
295	0.0001379
296	0.0001379
297	0.0001379
298	0.000135
299	0.000135
300	0.0001223

Table 4 shows the value of the Optimal solution at different iterations showing the first 10 and last 10 iterations.

Results

Makespan: 9.1509
Utilization Rate: 0.1093
Optimal Solution: 0.0001223

The obtained Optimal Solution after successful execution of 300 iterations as per parameters readings defined in Case 3 is shown in Figure 9. The X-axis denotes the number of iterations and Y-axis refers to the Optimal Solution.

Modified Firefly algorithm

The flowchart for the Modified Firefly algorithm has been depicted in Figure 10.

Algorithm

- Initialize the parameters and structure array.

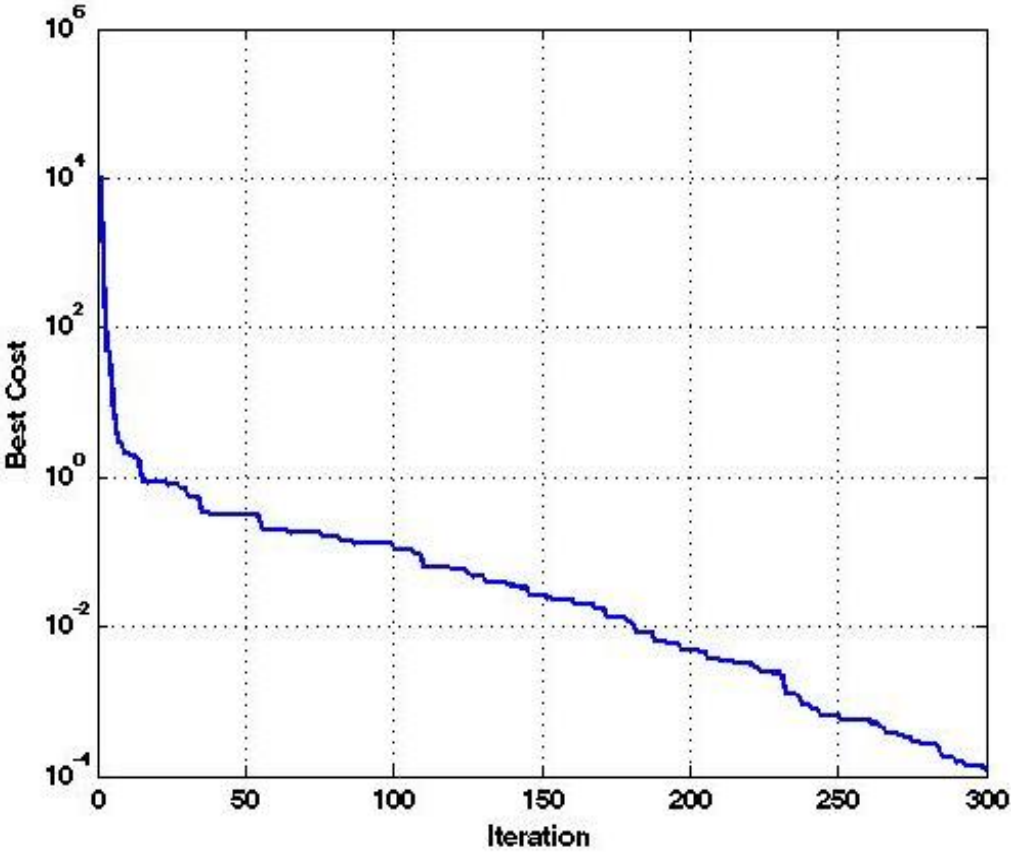

Figure 9: Plotted Graph Depicting Achieved Optimal Solution as per Case 3

- Populate the structure array with positions of the fireflies and set the initial cost to infinity.
- Initialize the number of pseudo time sets and reset the random generator.

  if nargin<1

  instr=[12 50];

  end

  n=instr(1);

  MaxGeneration=instr(2);

  rand('state',0);
- Generate the initial locations of n fireflies.

  [xn,yn,Lightn]=init_ffa(n,range);
- Initialize the range.
- Replicate and tile an array swarm.

  swarm=repmat(fly,n,1);
- Set global best cost gbest to infinity and initialize an array pos_gbest to hold global best positions.

  gbest=Inf;

  pos_gbest=[];
- Return an array of random numbers chosen from the continuous uniform distribution uisnf UNIFRND, swarm(i).pos.

  for i=1:n

  swarm(i).pos=unifrnd(smin,smax,1,d);

  swarm(i).cost=cost_func(swarm(i).pos);

  end
- Execute loop of n fireflies assigning swarm(i).pos to swarm(i).cost using cost function cost_func.
- Assigm swarm(i).cost to gbest and swarm(i).pos to pos_gbest as long as swarm(i).cost<gbest.

  if swarm(i).cost<gbest

  gbest=swarm(i).cost;

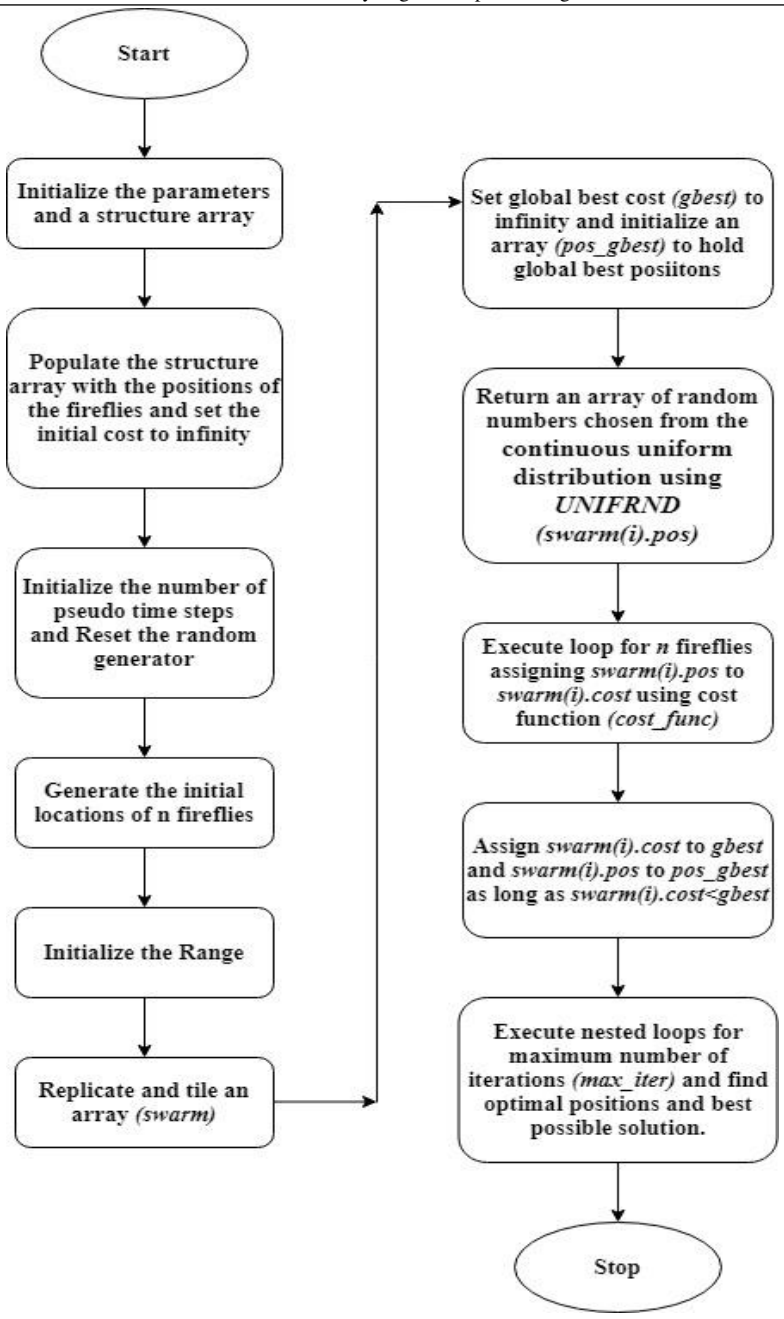

Figure 10: Flowchart for Modified Firefly algorithm

pos_gbest=swarm(i).pos;

end

- Execute nested loop for the maximum number of iterations max_iter and obtain results for optimal position and best possible solution.

Execution Parameters

The different parameters playing part in the execution of the Modified Firefly algorithm are stated below: -

Number of Decision Variables - d
Minimum value of Decision Variables - sMin
Maximum value of Decision Variables -sMax
Maximum Number of Iterations -

Table 5: Optimal Solution for iterations obtained as per specifications of Case 4

Iteration No.	Optimal Solution
1	1.4727
2	2.7405
3	2.7405
4	2.8976
5	2.8976
6	2.8976
7	2.8976
8	3.0069
9	3.0069
10	3.0069
CONTINUED	
291	2.9983
292	2.9983
293	2.9983
294	2.9983
295	2.9983
296	2.9983
297	2.9983
298	2.9983
299	2.9983
300	2.9999

Max_Iter Number of Fireflies (Swarm Size) - n
Light Absorption Coefficient - gamma
Attraction Coefficient Base Value - beta0
Mutation Coefficient - alpha
Mutation Coefficient Damping Ratio - damp

Case 4:

The values assigned to the participating parameters are depicted below.

d = 2
sMin = -10
sMax = 10
Max_Iter = 300
n = 25
gamma = 1
beta =1
alpha = 2 damp = 0.99

Table 5 shows the value of the Optimal Solution at different iterations, showing the first 10 and last 10 iterations.

Results

Makespan: 7.2652
Utilization Rate: 0.1376
Optimal Solution: 2.9999

Figure 11 shows the output for the Griewank test function as per readings of Case 1MFF.

Table 6 shows the values of different performance evaluation parameters obtained for Firefly and Modified Firefly algorithms.

Table 6: Readings of different performance evaluation parameters for Firefly and Modified Firefly

Algorithms	Makespan	Utilization Rate	Optimal Solution
Firefly	9.1509	0.1093	0.0001223
Modified Firefly	7.2652	0.1376	2.9999

The readings for different parameters for Firefly and Modified Firefly algorithms have been obtained in Table 6. The Makespan for Firefly algorithm has been recorded as 9.1509 and the Modified Firefly algorithm has been recorded as 7.2652. The lower is the value of Makespan, the better is the performance of the algorithm. The Utilization Rate is inversely proportional to the Makespan and therefore Modified Firefly algorithm has a higher Utilization Rate of 0.1376 as compared to the Firefly algorithm having a Utilization Rate of 0.1093. The higher value of the Utilization Rate indicates the optimized use of available resources. The optimal solution for the Modified Firefly algorithm is 2.9999 as compared to the Firefly algorithm which is .0001223.

Different fragments of Figure 12 show the comparison between Firefly and Modified Firefly algorithm for different performance evaluation parameters. In all the fragments, the X-axis denotes the name of the algorithms under consideration. The Y-axis represents the readings of different performance evaluation parameters. In fig. 12 (a), the Y-axis represents the reading of makespan, in Fig. 12 (b), Y-axis represents the utilization rate, in Fig. 12 (c) represents the readings for optimal value. The graphs indicate the superiority of the Modified Firefly algorithm in the case of all three performance evaluation parameters.

3.3 Fault-Tolerant Hybrid GWOFF algorithm

Figure 13 shows the flowchart for the proposed fault-tolerant hybrid GWOFF algorithm.

Algorithm

Initiate by entering the number of iterations.

1. Initialize the parameters relevant to GWO (Number of Decision Variables, Decision Variables Matrix Size, Decision Variables Lower and upper bound, Maximum Number of Iterations, Number of Fireflies (Swarm Size), Light Absorption Coefficient, Attraction Coefficient Base Value, Mutation Coefficient, Mutation Coefficient Damping Ratio, and Uniform Mutation Range).

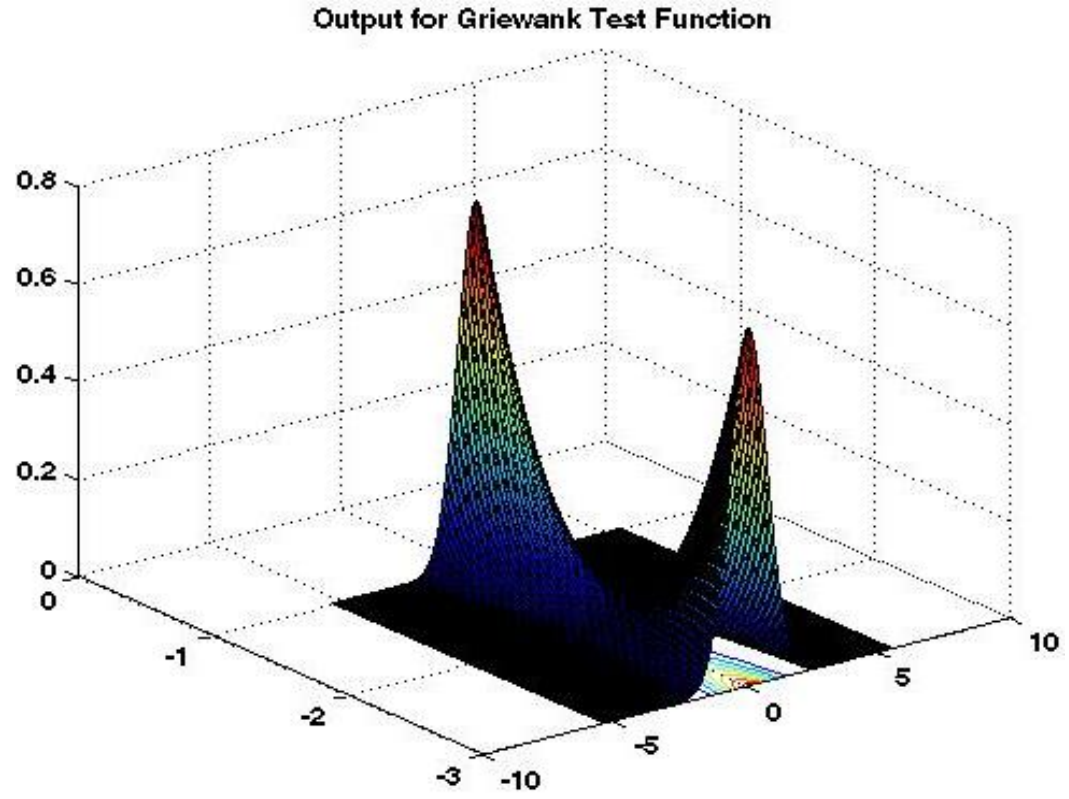

Figure 11: The Griewank Test Function Graph for Modified Firefly as per readings of Case 4

2. Select the benchmark function and load its details.
3. Create the search space and objective space.
4. Initialize the position of search agents.
5. Generate the initial population randomly to calculate the fitness of alpha, beta, and delta.
6. Return the search agents crossing the boundaries of the search space.
7. Evaluate the objective function for each search agent and update the values of alpha, beta, and delta.
8. Initiate the parameters related to the Firefly algorithm and define the initial cost function.
9. Populate the structure array with positions of the fireflies and set the initial cost to infinity.
10. Initialize the number of pseudo time sets and reset the random generator.
11. Generate the initial locations of n fireflies. [xn,yn,Lightn]=init_ffa(n,d);
12. Initialize the range.
13. Replicate and tile an array swarm. swarm=repmat(fly,n,1);
14. Set global best cost gbest to infinity and initialize an array pos_gbest to hold global best positions.

    gbest=Inf;

    pos_gbest=[];
15. Return an array of random numbers chosen from the continuous uniform distribution using UNIFRND, swarm(i).pos.

    for i=1:n

    swarm(i).pos=unifrnd(smin,smax,1,d);

    warm(i).cost=cost_func(swarm(i).pos);

    end
16. Execute loop of n fireflies assigning swarm(i).pos to swarm(i).cost using cost function cost_func.
17. Assigm swarm(i).cost to gbest and swarm(i).pos to pos_gbest, as long as swarm(i).cost<gbest.

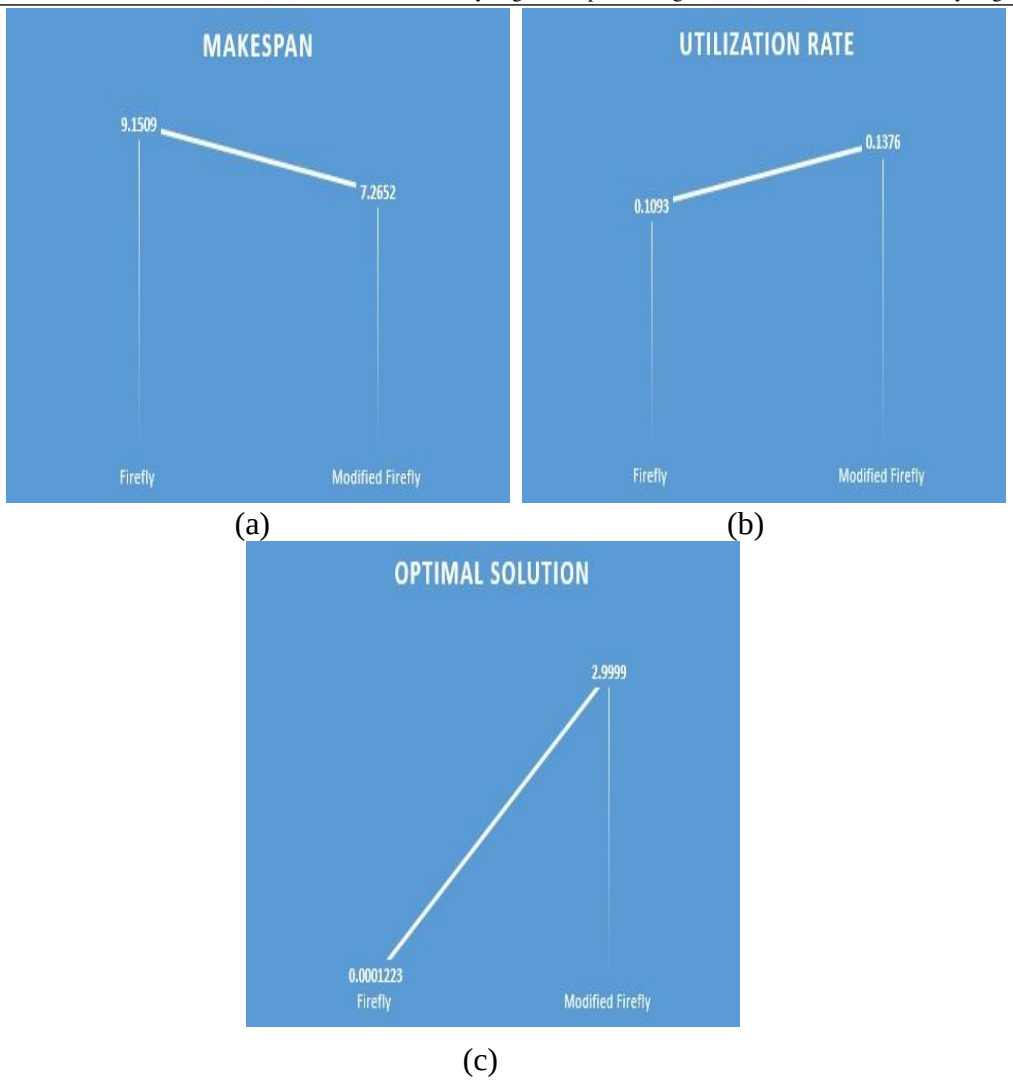

Figure 12: (a) Makespan, (b) Utilization Rate, (c) Optimal Solution

if swarm(i).cost<gbest

gbest=swarm(i).cost;

pos_gbest=swarm(i).pos;

end

18. Execute nested loop for the maximum number of iterations max_iter and obtain results for optimal position and best possible solution.
19. Call Firefly routine and get updated positions.
20. Enter the number of Virtual Machines (VMs) and faulty agent scenarios with the number of faulty agents in each scenario.
21. Set the lower limit of faulty agents to 0 in each scenario and enter the upper limit dynamically.
22. Calculate the maximum value of the best fitness function.
23. If current iteration < Maximum number of iterations

    Go to Step 2

    Else

    Go to Step 24
24. End the process.

Execution Parameters

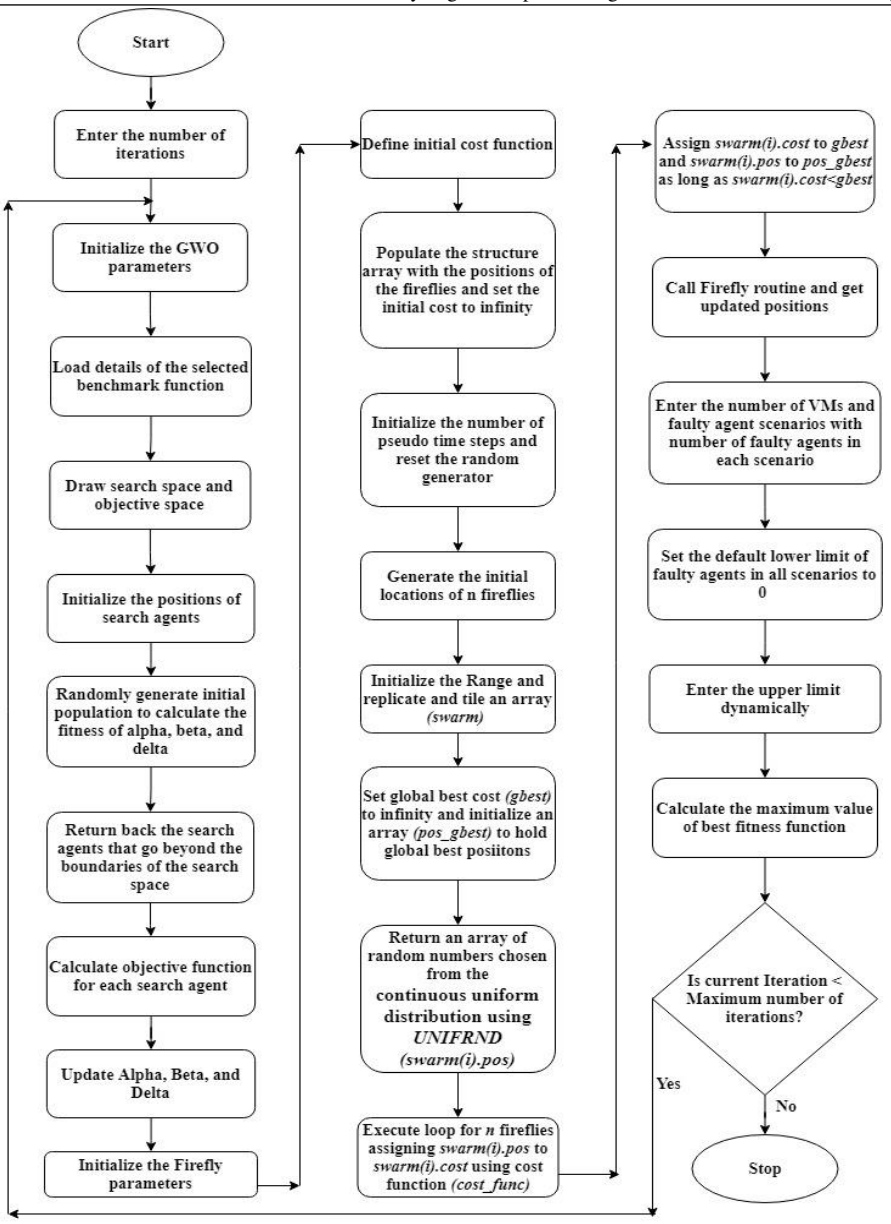

Figure 13: Flowchart of the Proposed Fault-Tolerant Hybrid GWOFF algorithm

Beyond the execution parameters involved in Modified versions of GWO and FF, the following parameters participated in the execution of the fault-tolerant hybrid GWOFF algorithm.

Number of iterations - iter
Number of virtual machines - np
Number of faulty agents scenario - dim
Number of faulty agents in each scenario - dmat(1,i)
Case 5:
Iter - 300
Np - 10
Dim - 3
Number of faulty agents in Scenario 1= 50
Number of faulty agents in Scenario 2= 10
Number of faulty agents in Scenario 3= 50

Results

The max value of the fitness function= 2.981882 With Fault Tolerance
Makespan: 8.9684
Utilization Rate: 0.1115

Turnaround Time (deci seconds): 10.6351
Without Fault Tolerance
Makespan: 10.4605
Utilization Rate: 0.0956
Turnaround Time (deci seconds): 12.1272

Figure 14 shows the final positions of the agents at the end of the 300th iteration.

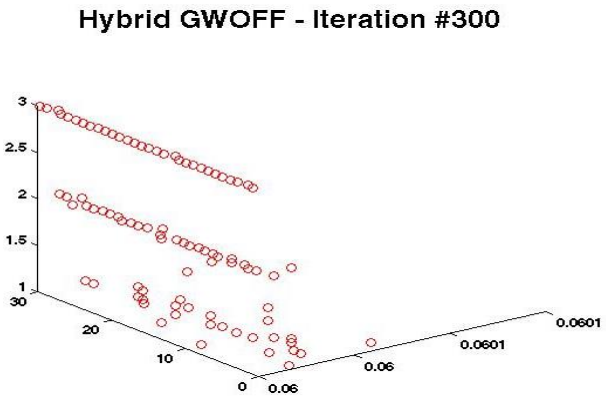

Figure 14: The Final Positions of the agents at the end of the 300th iteration as per Case 1Fault Tolerance Hybrid

Figure 15 shows the performance of the proposed Hybrid GWOFF algorithm without fault tolerance red color) and with fault tolerance (green color). The variation in the red-colored graph is much more as compared to the variation in the green-colored graph. This indicates that the stability in the case of a Hybrid GWOFF algorithm with fault-tolerance is much more as compared to the one without fault-tolerance.

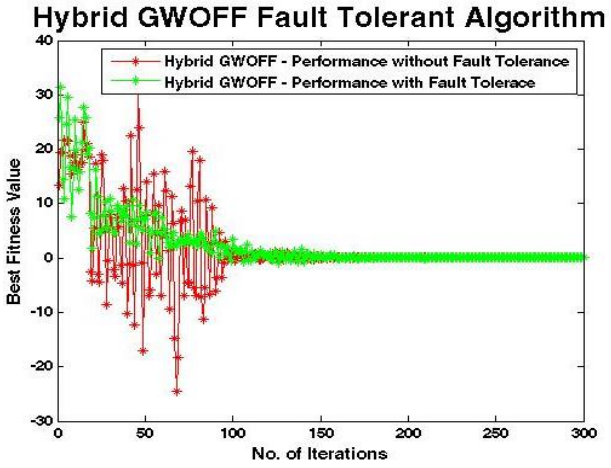

Figure 15: Performance of Hybrid GWOFF without and with fault tolerance as per Case 1 Fault Tolerance Hybrid

Table 7: Results of K-means clustering with K=6 for the survey dataset

Scenario	Makespan	Utilization Rate	Turnaround Time
With Fault Tolerance	8.9684	0.1115	10.6351
Without Fault Tolerance	10.4605	0.0956	12.1272

Table 7 shows the readings of different performance evaluation parameters obtained after executing the Hybrid GWOFF algorithm as per specifications of Case 5.

Different fragments of Figure 16 show the comparison between the Hybrid GWOFF algorithm with and without fault tolerance for different performance evaluation parameters. In all the fragments, the X-axis denotes the name of the algorithms under consideration. The Y-axis represents the readings of different performance evaluation parameters. In Fig. 16 (a), the Y-axis represents the reading of makespan, in Fig. 16 (b), Y-axis represents the utilization rate, and in Fig. 16 (c) represents denotes the turnaround time in deci seconds. The graphs indicate the superiority of the Hybrid GWOFF algorithm with fault tolerance for all three performance evaluation parameters.

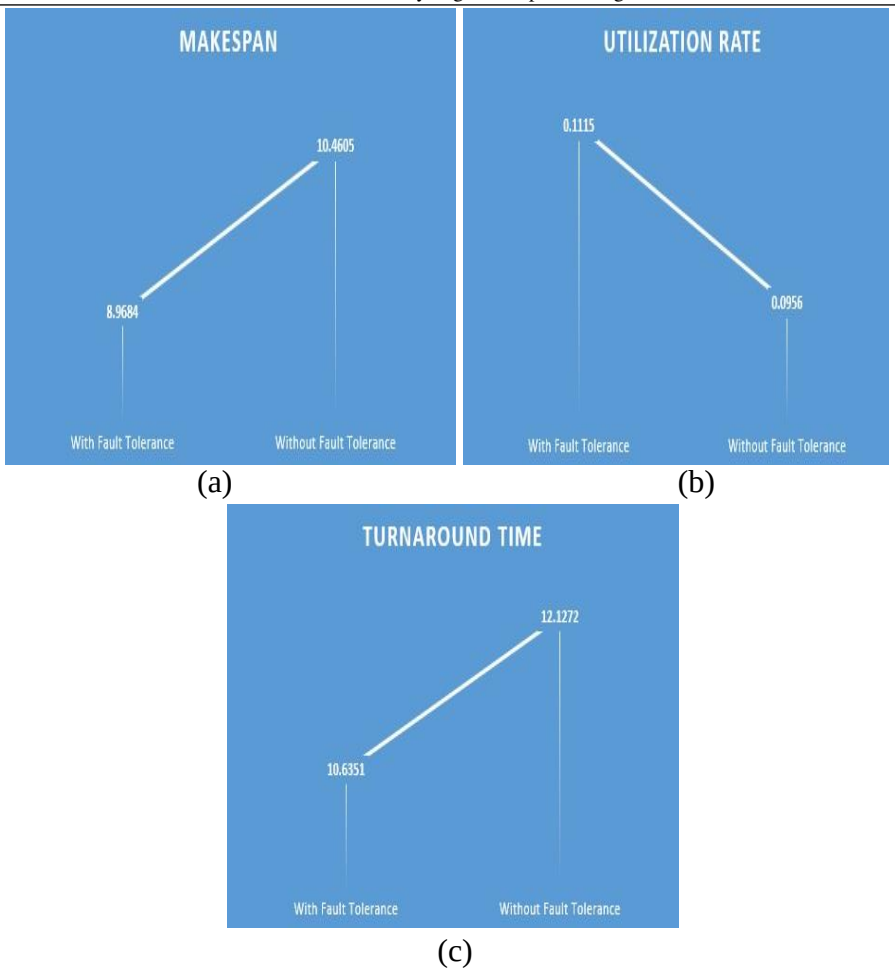

Figure 16: (a) Makespan, (b) Utilization Rate, (c) Turnaround Time

4 Conclusion

In the case of the Grey Wolf Optimization algorithm, the Makespan for GWO has been recorded higher than the Hybrid GWOPSO. Hybrid GWOPSO has a higher Utilization Rate as compared to GWO. The Throughput for Hybrid GWOPSO is greater than GWO. The Turnaround Time for Hybrid GWOPSO is on the lower side as compared to that of GWO. The Waiting Time of GWO is much higher as compared to Hybrid GWOPSO. The Optimal Value for Hybrid GWOPSO is higher than GWO. The greater value obtained for Optimal Value for Hybrid GWOPSO indicates the superiority of Hybrid GWOPSO over GWO. In the case of Firefly algorithms, the Makespan for Modified Firefly has been recorded lower than Conventional Firefly. The Modified Firefly has a higher Utilization Rate as compared to Conventional Firefly. In the case of the Hybrid GWOFF algorithm, the performance of the algorithm is with higher stability as compared to the one without fault tolerance. The experimental results of the implemented Hybrid GWOFF Fault Tolerant algorithm show an improvement in performance over the Hybrid GWOFF without Fault Tolerance. The performance evaluation of the algorithms has been measured in terms of the three main scheduling performance metrics: Makespan, Utilization Rate, and Turnaround time. The readings of Makespan and Turnaround time have been found lower in the case of Hybrid GWOFF with fault tolerance under both the scenarios tested with varying numbers of iterations, VMs, and Faulty Scenarios. The reading of Utilization Rate has been found higher in the case of Hybrid GWOFF with fault tolerance under both the scenarios tested with varying numbers of iterations, VMs, and Faulty Scenarios. This indicates that the performance of Hybrid GWOFF with fault tolerance is much better than the one without fault tolerance. In the future, the number of per-

formance evaluation parameters could be increased and the hybridization makes take place among more than two meta-heuristic algorithms. Although improvements have been witnessed in the optimization algorithms using swarm intelligence and meta-heuristics algorithms, yet a unique algorithm capable of handling all optimization problems effectively and with the best parameters readings is to come up. The development process of hybrid algorithms will continue until the best combination of algorithms would be found.

References

[1] Binitha, S., Sathya, S. S., et al. A survey of bio inspired optimization algorithms. *International journal of soft computing and engineering*, 2(2):137–151, 2012.

[2] Chou, J.-S. and Ngo, N.-T. Modified firefly algorithm for multidimensional optimization in structural design problems. *Structural and Multidisciplinary Optimization*, 55(6):2013–2028, 2017.

[3] El-Gaafary, A. A., Mohamed, Y. S., Hemeida, A. M., and Mohamed, A.-A. A. Grey wolf optimization for multi input multi output system. *Universal Journal of Communications and Network*, 3(1):1–6, 2015.

[4] Fister Jr, I., Yang, X.-S., Fister, I., and Brest, J. Memetic firefly algorithm for combinatorial optimization. *arXiv preprint arXiv:1204.5165*, 2012.

[5] Gupta, S. and Deep, K. A novel random walk grey wolf optimizer. *Swarm and evolutionary computation*, 44:101–112, 2019.

[6] Hatta, N., Zain, A. M., Sallehuddin, R., Shayfull, Z., and Yusoff, Y. Recent studies on optimisation method of grey wolf optimiser (gwo): a review (2014–2017). *Artificial Intelligence Review*, 52(4):2651–2683, 2019.

[7] Heidari, A. A. and Pahlavani, P. An efficient modified grey wolf optimizer with lévy flight for optimization tasks. *Applied Soft Computing*, 60:115–134, 2017.

[8] Hu, P., Chen, S., Huang, H., Zhang, G., and Liu, L. Improved alpha-guide grey wolf optimizer. *IEEE Access*, 7:54215437, 2019.

[9] Ibrahim, R. A., Abd Elaziz, M., and Lu, S. Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization. *Expert Systems with Applications*, 108:1–27, 2018.

[10] Kohli, M. and Arora, S. Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of computational design and engineering*, 5(4):458–472, 2018.

[11] Kumar, V. and Kumar, D. An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems. *Advances in Engineering Software*, 112:231–254, 2017.

[12] Liu, Y., Sun, J., Yu, H., Wang, Y., and Zhou, X. An improved grey wolf optimizer based on differential evolution and otsu algorithm. *Applied Sciences*, 10(18):6343, 2020.

[13] Long, W., Jiao, J., Liang, X., and Tang, M. Inspired grey wolf optimizer for solving large-scale function optimization problems. *Applied Mathematical Modelling*, 60:112–126, 2018.

[14] Long, W., Liang, X., Cai, S., Jiao, J., and Zhang, W. A modified augmented lagrangian with improved grey wolf optimization to constrained optimization problems. *Neural Computing and Applications*, 28(1):421–438, 2017.

[15] Long, W., Wu, T., Cai, S., Liang, X., Jiao, J., and Xu, M. A novel grey wolf optimizer algorithm with refraction learning. *IEEE Access*, 7:57805–57819, 2019.

[16] Memari, A., Ahmad, R., and Rahim, A. R. A. Metaheuristic algorithms: guidelines for implementation. *Journal of Soft Computing and Decision Support Systems*, 4(6):1–6, 2017.

[17] Nasrabadi, M. S., Sharafi, Y., and Tayari, M. A parallel grey wolf optimizer combined with opposition based learning. In *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 18–23. IEEE, 2016.

[18] Niu, P., Niu, S., Chang, L., et al. The defect of the grey wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, 171:37–43, 2019.

[19] Singh, N. and Singh, S. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *Journal of Applied Mathematics*, 2017, 2017.

[20] Wahid, F. and Ghazali, R. Hybrid of firefly algorithm and pattern search for solving optimization problems. *Evolutionary Intelligence*, 12(1):1–10, 2019.

[21] Wang, G.-G., Guo, L., Duan, H., and Wang, H. A new improved firefly algorithm for global numerical optimization. *Journal of Computational and Theoretical Nanoscience*, 11(2):477–485, 2014.

[22] Wang, H., Wang, W., Sun, H., and Rahnamayan, S. Firefly algorithm with random attraction. *International Journal of Bio-Inspired Computation*, 8(1):33–41, 2016.

[23] Yang, X.-S. and He, X. Firefly algorithm: recent advances and applications. *International journal of swarm intelligence*, 1(1):36–50, 2013.

[24] Yu, S., Zhu, S., Ma, Y., and Mao, D. A variable step size firefly algorithm for numerical optimization. *Applied Mathematics and Computation*, 263:214–220, 2015.

[25] Yue, Z., Zhang, S., and Xiao, W. A novel hybrid algorithm based on grey wolf optimizer and fireworks algorithm. *Sensors*, 20(7):2147, 2020.

[26] Zhang, S. and Zhou, Y. Grey wolf optimizer based on powell local optimization method for clustering analysis. *Discrete Dynamics in Nature and Society*, 2015, 2015.