Improved Multi–Objective Firefly Algorithms to Find Optimal Golomb Ruler Sequences for Optimal Golomb Ruler Channel Allocation

Shonak Bansal, Prince Jain, Arun Kumar Singh, Neena Gupta

Abstract—Recently nature–inspired algorithms have widespread use throughout the tough and time consuming multi–objective scientific and engineering design optimization problems. In this paper, we present extended forms of firefly algorithm to find optimal Golomb ruler (OGR) sequences. The OGRs have their one of the major application as unequally spaced channel–allocation algorithm in optical wavelength division multiplexing (WDM) systems in order to minimize the adverse four–wave mixing (FWM) crosstalk effect. The simulation results conclude that the proposed optimization algorithm has superior performance compared to the existing conventional computing and nature–inspired optimization algorithms to find OGRs in terms of ruler length, total optical channel bandwidth and computation time.

Keywords—Channel allocation, conventional computing, four–wave mixing, nature–inspired algorithm, optimal Golomb ruler, Lévy flight distribution, optimization, improved multi–objective Firefly algorithms, Pareto optimal.

I. INTRODUCTION

There exists a rich collection of adverse nonlinear optical effects [1], [2] that degrade the performance of optical WDM systems. Out of these nonlinearities, the performance degradation by FWM crosstalk is a serious problem for WDM systems and can be minimized by unequal channel spacing concept [1], [2]. To minimize the FWM crosstalk effects in optical WDM systems, numerous unequally spaced channel allocation (USCA) algorithms [1], [3], [4] have been proposed, but have the drawback of increased optical bandwidth requirement. In order to minimize FWM crosstalk effects, this paper proposes an USCA algorithm based on OGR sequences [5], [6].

Golomb rulers are an ordered set of non–negative integer locations \( a_i \) such that all the positive differences \( a_j - a_i \) are distinct [7], [8]. These non–negative integer locations are referred to as marks [7], [9]. An OGR is the shortest length ruler for a given number of marks [10], [11]. Multiple different OGRs can exist for a specific number of marks. By using OGRs in optical WDM systems, it is possible to achieve the smallest dissimilar number to be used for the optical WDM channel–allocation problem. As the difference between any two numbers is different, the new FWM frequency signals generated would not fall into the one already assigned for the carrier channels. According to the literatures [7], [12], [13], Golomb rulers represent a class of NP–complete problems. For higher order marks, the exhaustive computer search [14], [15] of such NP–complete problems is difficult. There are numerous algorithms [14]-[16] to solve such a problem. To date, no better algorithm is known for finding the minimum length ruler. Numerous nature–inspired optimization algorithms and their hybridization have been efficiently realized in [9], [16]-[27] to solve such NP–complete problems that provide a good starting point for algorithms of finding OGR sequences. But, by using OGRs in optical WDM systems as an unequally spaced channel–allocation algorithm, there are two objectives (bi–objective) i.e. optimal ruler length and optimal total channel bandwidth. This paper presents the application of modified forms of Multi–objective Firefly algorithm (MOFA), to find either optimal or near–optimal rulers in a reasonable time and its performance comparison with the existing conventional i.e. Extended quadratic congruence (EQC) [1], [4] and Search algorithm (SA) [1], [4] and nature–inspired i.e. Genetic algorithm (GA) [9], and its simple form MOFA [22], [23] to find OGRs up to several order marks for WDM systems. The improvement in the performance of MOFA is performed by introducing the concept of random walk i.e. Lévy flight distribution [28], differential evolution mutation strategy [29] and parallelism. To our best knowledge, these improvements have not been implemented to find OGRs.

II. MODIFIED MOFAS

Due to highly nonlinearity and complexity, optimization in engineering design fields tends to be very tough and challenging. Since the use of conventional or exact algorithms for finding optimal solutions to a multi–objective engineering design problem is impractical in terms of computational resources, so they are not best tools for global optimization. Nature–inspired multi–objective algorithms are very dominant in dealing with optimization design problems [30].

This section introduces the modified forms of MOFA [31]. Inspired by the flashing pattern and characteristics of fireflies, by using three idealized rules, Yang [32] developed an algorithm called Firefly algorithm (FA) for the optimization of single objective and extended it to solve multi–objective problems [31]. In FA, the variation of light intensity \( I \) and the formulation of attractiveness \( \beta \) with distance \( r \) between any two fireflies are two main issues [32]. For a given medium having a
fixed light absorption coefficient $\gamma$, the movement of a firefly $i$ is attracted to another brighter firefly $j$ is:

$$ X_i = X_i + \beta_0 e^{-\frac{\gamma}{s_{ij}^2}} (X_j - X_i) + \alpha \text{rand} $$

(1)

where $I_0$ is original light intensity and $\beta_0$ is attractiveness at $r = 0$, $s_{ij}$ is Cartesian distance [32] between any two fireflies $i$ and $j$ at locations $X_i$ and $X_j$, respectively. The second term in (1) is due to attraction and the third term is randomization with a control parameter $\alpha$. For most cases in the implementation, $\beta_0 = 1$ and $\alpha \epsilon [0, 1]$.

The MOFA uses the same equations and idealized rule as for FA for optimizing the multiple objectives. In MOFA, a design problem with $L$ individual objective functions, nonlinear equality and inequality constraints are combined into a single composite function by using weighted sum method [31]:

$$ f = \sum_{i=1}^{L} w_i f_i \text{ with } \sum_{i=1}^{L} w_i = 1, \quad w_i > 0, $$

(2)

where $w_i$ are randomly generated non-negative weights that act as preferences for optimizing the multi–objectives, so that the Pareto optimal front [30], [31] can be approximated correctly.

The success of nature–inspired optimization algorithms lies in how fast the algorithms explore the new possible solutions and how efficiently they exploit the better solutions. Although MOFA in its simplified form works well in the exploitation, there are still some problems in global exploration of the search space [33] because of the phenomenon of low accuracy and slow convergence rate. If all solutions in the initial phase of the algorithm are collected in a small part of search space, the algorithm may not find the optimal result and with a high probability, it may be trapped in that sub–domain. One can consider a large number of solutions to avoid this shortcoming, but it causes an increase in the function calculations as well as the computational costs and time. So for MOFA, there is a need by which exploration and exploitation can be enhanced.

To enhance the performance of MOFA, two features, Lévy–flight distribution [28] and fitness (cost) value based differential evolution mutation strategy [29] to explore search space are introduced in the algorithm. Further to exploit the search space, the parallelism concept based on multiple populations is introduced to validate MOFA performance with and without Lévy–flight and mutation strategy. The Lévy flight distribution is:

$$ L(\lambda) \sim \frac{\Gamma(\lambda)}{\pi} \sin (\pi \lambda / 2) \frac{1}{s(\lambda) \sqrt{s}}, \quad (s >> s_0 > 0) $$

(3)

$\Gamma(\lambda)$ is gamma distribution valid for large steps i.e. for $s > 0$. Throughout the paper, $\lambda = 1.5$ is used. In theory, it is required that $|s_0| > 0$, but in practice $s_0$ can be as small as 0.1.

By combining the characteristics of Lévy flights with the MOFA, another new algorithm named, Lévy flight multi–objective Firefly algorithm (LMOFA) can be formulated. For LMOFA, the third term in (1) is randomized via Lévy flights. The firefly movement equation for LMOFA is [33]:

$$ X_i = X_i + \beta_0 e^{-\frac{\gamma}{s_{ij}^2}} (X_j - X_i) + a.\text{sign}(\text{rand} - 0.5) \otimes L(\lambda) $$

(4)

The product $\otimes$ means the entry wise multiplications. The term $\text{sign}(\text{rand} - 0.5)$, where $\text{rand} \epsilon [0,1]$ essentially provides a random direction, while the random step length is drawn from a Lévy distribution $L(\lambda)$.

The mutation rate probability $MR^i$ related to the fitness value $f_i^t$ of each solution $x_i$ and maximum fitness value Max($f^t$) in the population at running iteration index $i$ is:

$$ MR^i = \frac{f_i^t}{\text{Max}(f^t)} $$

(5)

Instead of using the fixed DE mutation operator, this paper uses the varying mutation operators at running iteration $t$:

$$ F^t_1 = \left( UB - LB \right) \frac{f_i^t}{\eta} + UB $$

$$ F^t_2 = \left( UB - LB \right) \frac{f_i^t}{\eta} + LB $$

(6)

where $LB$, $UB$ are lower and upper bound on the solutions respectively, $\beta_1$, $\beta_2 \epsilon [0,1]$ are random vectors drawn from uniform distribution, and $\eta$ is positive fixed parameter with large values. In order to make mutation operators $F^t_1$ and $F^t_2$ less than unity, the values of $\beta_1$, $\beta_2$ and $\eta$ are selected carefully. In simplest case,

$$ \beta_1 = \text{rand}_1 \cdot 0.0001 \text{ and } \beta_2 = \text{rand}_2 \cdot 0.0001 $$

(7)

$$ \eta = 2 \cdot \text{maximum number of iterations} $$

(8)

where $\text{rand}_1$ and $\text{rand}_2$ are random numbers between $[0,1]$. The mutation equation used in this paper is:

$$ x_i^{t+1} = x_i^t + F_1^t (s_{\text{best}}^t - x_i^t) + F_2^t (x_i^t - r_2^t) $$

(9)

where $s_{\text{best}}^t$ is population at iteration index $t$, $s_{\text{best}}^t$ is the current global best solution at iteration index $t$, $r_1$ and $r_2$ are uniformly distributed random integer numbers between 1 to problem size. The numbers $r_1$ and $r_2$ are different from running index. If mutation strategy is combined with MOFA and LMOFA, MOFA with mutation (MOFAM) and LMOFA with mutation (LMOFAM) can be formulated. If multiple populations (parallelism) are introduced with MOFAM and LMOFAM, then other novel algorithms, namely PLMOFA and PLMOFAM can be formulated.

The corresponding pseudo–code for improved MOFA (I–MOFA) is shown in Fig. 1. Noted that I–MOFA presents the pseudo–code for PLMOFAM.
Begin
/* parameter initialization */
Define objective functions \( f_1(x), \ldots, f_L(x), \quad x = (x_1, \ldots, x_d) \);
/* d is the dimension of the problem */
Generate initial fireflies of MP populations each of size \( NP \)
\((i = 1, 2, \ldots, MP; \text{ and } j = 1, 2, \ldots, NP)\);
/* MP is multi–parallel/entire population size and \( NP \) is size of sub–populations in \( MP \)*/
Define light absorption coefficient \( \gamma \);
For \( i = 1 \): MP
For \( j = 1 \): NP
Generate \( L \) weights \( w_j \) so that \( \sum_{j=1}^{L} w_j = 1 \) and form a single
objective i.e. light intensity \( I_e \); Find the local best among \( i \)th population of \( NP \) fireflies;
End for
End for
Based on fitness value, among \( MP \) solutions select globally
best solution \( x^* \);
/* End of parameter initialization */
For \( i = 1 : N \)
/* \( N \) is the Pareto fronts points */
Generate \( L \) weights which satisfies (2);
While not \( TC \)
/* \( TC \) is the termination criterion */
For \( j = 1 : MP \)
For \( k = 1 : NP \)
/* all \( NP \) fireflies*/
For \( m = 1 : 1; k \)
If \( I_{kj}^e > I_{kj}^f \)
Move firefly \( k \) towards \( m \) via Lévy flights;
End if
/* Mutation */
Compute mutation rate probability \( MR \); If \( (MR < \text{rand}(0,1)) \)
Perform mutation;
End if
/* End of mutation */
Vary attractiveness with distance \( r \) via \( \exp[-\gamma r] \); Evaluate new generated \( NP \) solutions of \( i \)th population;
Form single optimize objective to update light intensity;
Rank the solutions and find current best Pareto optimal
solution \( x_{i\text{best},m} \);
End for
End for
/* all \( MP \) */
Find global best solution \( x^r \) among \( MP \times x_{i\text{best}} \) solutions;
End while
Record \( x \) as a non–dominated solution;
End for
Postprocess results and visualization;
End

Fig. 1 Pseudo–code for I–MOFA

By removing the concept of parallelism, Fig. 1 represents the pseudo–code for LMOFAM, if both the concept of parallelism and mutation are omitted from Fig. 1, then it represents the pseudo–code for MOFAM if the concept of only mutation is omitted then it represents the pseudo–code for PLMOFA, and if the concept of mutation and Lévy flights are omitted then it corresponds to the pseudo–code for PMOFAM

III. FINDING OGRs

If the spacing between any pair of channels in Golomb ruler set is denoted as \( CS \), an individual element i.e. non–negative integer location is as \( IE \) and the total number of channels as \( n \), then the two optimization objectives \( f_1(x) = \text{ruler length} (RL) \) and \( f_2(x) = \text{total optical channel bandwidth} (TBW) \) are [9]:

\[
RL = \sum_{i=1}^{n} (CS) \quad \text{subject to } (CS)_i \neq (CS)_j \quad (10)
\]

\[
TBW = \sum_{i=1}^{n} (IE) \quad \text{subject to } (IE)_i \neq (IE)_j \quad (11)
\]

where \( i, j = 1, 2, \ldots, n \) with \( i \neq j \) are distinct in (10) and (11). The objectives \( f_1(x) \) and \( f_2(x) \) are combined into a single composite objective \( f(x) \). The proposed general pseudo–code for the improved MOFA forms with Lévy–flight, fitness value based DE mutation strategy and parallelism to find OGRs for WDM systems is shown in Fig. 2.

IV. SIMULATION RESULTS

This section presents the performance of proposed MOFA algorithms and their performance comparison with two existing conventional algorithms and two nature–inspired algorithms of finding unequal channel spacing. The algorithms have been coded and tested in MATLAB language running under Windows 7, 64–bit operating system and were run 20 times to obtain OGRs.

A. Simulation Parameters Selection for I–MOFA’s

To find the optimal sequences, the best parameter values for proposed algorithms finally settled are shown in Table I where \( n \) denotes the number of channels/ marks. The parameters multi–parallel population size (M–Popsize), Sub–population size (S–Popsize), \( \eta \), and Pareto front points (N) are not required by algorithms LMOFA, LMOFAM, and MOFAM, so they are shown by a dash line. The maximum number of iterations \( \text{(Maxiter)} \) set for all algorithms is number of marks times 100. By introducing parallelism in MOFA and hybridization of parallel MOFA with Lévy flights and mutation strategy the algorithm finds OGRs in less number of iterations due to which the computation time is optimized as there is exploration and exploitation of search space. This means that the performance of algorithm is enhanced.

B. Comparison of Proposed Algorithms with Previous Existing Algorithms in Terms of Ruler Length and Total Optical Channel Bandwidth

The ruler length and total occupied channel bandwidth for different sequences obtained by the proposed improved forms of MOFA after 20 executions and their performance comparison with EQC, SA, GAs, and MOFA are reported in Table II. The applications of EQC and SA are restricted to prime powers only, so the ruler length and total occupied channel bandwidth for EQC and SA are presented by a dash line in Table II [1]. Fig. 3 illustrates the graphical representation of Table II. Comparing simulation results obtained from the proposed algorithms with the existing algorithms, it is noted that there is significant improvement in the ruler length and hence the total occupied channel bandwidth. This improvement is due to the better accuracy and
fast convergence rates illustrated by introducing the concept of random walk by Lévy flight distribution, DE mutation strategy and multi–population with MOFA. From Table II, it is also noted that the algorithms LMOFA and MOFAM can find the shortest length rulers up to 15–channels, LMOFAM up to 16–channels, whereas PLMOFA and PLMOFAM up to 20–channels efficiently. The maximum numbers of iterations required by the algorithms LMOFA, MOFAM, LMOFAM, PLMOFA and PLMOFAM for 20–channel Golomb ruler are 1800, 1600, 1200, 800 and 500 respectively.

Table I

<table>
<thead>
<tr>
<th>Parameter</th>
<th>LMOFA</th>
<th>LMOFAM</th>
<th>MOFAM</th>
<th>PLMOFA</th>
<th>PLMOFAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi–parallel population size (M–Popsize)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Sub–population size (S–Popsize)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Size of entire search space (Popsize)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>n*100</td>
<td>n*100</td>
</tr>
<tr>
<td>Maximum Iteration (Maxiter)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>n*100</td>
<td>2 * Maxiter</td>
</tr>
<tr>
<td>Pareto front points (N)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>α</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>β</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>γ</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Begin

/* Parameter initialization */
Initialize the number of channels n, upper bound on the ruler length and Pareto fronts point N;
Define light absorption coefficient γ;
Generate a set of MP integer populations (fireflies) each of size NP integers randomly and each integer NP population corresponding to Golomb ruler to the specified channels; /* Number of integers in firefly is being equal to the number of channels */
For i = 1 : MP
For j = 1 : NP
Find the local best j刹那x jlbestx, among ith population of NP fireflies by using (2), (10) and (11);
End for
End for
Based on fitness value (Light intensity I), among MP lbestx solutions select the globally best solution x*;
/* End of parameter initialization */
For i = 1 : N
Generate L weights which satisfies (2);
While not TC /* TC is termination criterion */
For j = 1 : MP
For k = 1 : NP /*All NP fireflies*/
For m = 1 : k
A: If I_m > I_k
Move firefly k towards m in d–dimension via Lévy flights;
End if
/* Mutation */
Based upon the mutation rate probability MR, perform mutation;
/* End of mutation */
Check Golombness of updated solutions;
If Golombness is satisfied
Retain that solution and then go to B;
Else
Retain the previous generated solution into the parameter initialization step and then go to A;
End if
B: Vary attractiveness with distance r via exp[−γr];
Evaluate new generated NP solutions of jth population and form a single optimize objective to update light intensity;
Rank the solutions and find current best Pareto optimal solution xj刹那,m刹那;
End for
End for
End while
Record x, as a non–dominated solution;
End for
Postprocess results and visualization;
End

Fig. 2 General Pseudo–code for I–MOFA to find OGRs for optical WDM systems
C. Comparison of Proposed Algorithms in Terms of Computational Time

Finding OGRs for higher order marks by exhaustive search algorithms are very time consuming, which means that it takes several hours, months, and even years of calculation on the network of several thousand computers [6], [7], [34], [35]. In [17], it is identified that to find Golomb ruler sequences from several hours, months, and even years of calculation on the algorithms are very time consuming, which means that it takes approximately 12.57 minutes for 10–marks, 2.28 years for 12–marks, 960 seconds for 11–marks, 1913 seconds for 10–marks, 960 seconds for 11–marks, 1913 seconds for 12–marks and 2516 seconds for 13–marks. In [20], it is reported that CPU time taken by Tabu search algorithm is around 0.1 second for 5–marks, 720 seconds for 10–marks, 960 seconds for 11–marks, 1913 seconds for 12–marks and 2516 seconds for 13–marks. The OGRs realized by hybrid GA [20] took around 5 hours for 11–marks, 8 hours for 12–marks, and 11 hours for 13–marks. The OGRs realized by the exhaustive search algorithms [15] for 14 and 16–marks, took nearly 1 hour and 100 hours respectively, while 17, 18 and 19–marks realized in [34], took around 1440, 8600 and 36200 CPU hours (nearly seven months) respectively on a Sun Sparc Classic workstation. Also, the near-OGRs realized up to 20–marks by GAs, the maximum execution time was approximately 31 hours, whereas for

| Table II | Comparison of Proposed Algorithms with Existing Conventional and Nature-Inspired Algorithms in Terms of Ruler Length and Total Bandwidth |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| **n** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** | **RL** | **TBW (Hz)** |
| 3 | 6 | 10 | 6 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 |
| 4 | 15 | 28 | 15 | 28 | 6 | 11 | 6 | 11 | 6 | 11 | 6 | 11 | 6 | 11 | 6 | 11 |
| 5 | — | — | — | — | 12 | 23 | 12 | 23 | 12 | 23 | 12 | 23 | 12 | 23 | 12 | 23 |
| 6 | 45 | 140 | 20 | 60 | 17 | 42 | 17 | 42 | 17 | 42 | 17 | 42 | 17 | 42 | 17 | 44 |
| 7 | — | — | — | — | 29 | 79 | 29 | 79 | 29 | 79 | 29 | 79 | 29 | 79 | 29 | 73 |
| 8 | 91 | 378 | 49 | 189 | 35 | 126 | 35 | 126 | 35 | 126 | 35 | 126 | 35 | 126 | 35 | 126 |
| 9 | — | — | — | — | 59 | 196 | 59 | 196 | 59 | 196 | 59 | 196 | 59 | 196 | 59 | 196 |
| 10 | — | — | — | — | 75 | 283 | 75 | 283 | 75 | 283 | 75 | 283 | 75 | 283 | 75 | 283 |
| 12 | 231 | 1441 | 132 | 682 | 123 | 532 | 123 | 532 | 123 | 532 | 123 | 532 | 123 | 532 | 123 | 532 |
| 13 | — | — | — | — | 137 | 650 | 137 | 650 | 137 | 650 | 137 | 650 | 137 | 650 | 137 | 650 |
| 14 | 325 | 2340 | 286 | 1820 | 228 | 1177 | 228 | 1177 | 228 | 1177 | 228 | 1177 | 228 | 1177 | 228 | 1177 |
| 16 | — | — | — | — | 298 | 1653 | 298 | 1653 | 298 | 1653 | 298 | 1653 | 298 | 1653 | 298 | 1653 |
| 18 | 561 | 5203 | 493 | 5100 | 427 | 2599 | 427 | 2599 | 427 | 2599 | 427 | 2599 | 427 | 2599 | 427 | 2599 |
| 19 | — | — | — | — | 597 | 5067 | 597 | 5067 | 597 | 5067 | 597 | 5067 | 597 | 5067 | 597 | 5067 |
| 20 | 703 | 7163 | 703 | 6460 | 673 | 4826 | 673 | 4826 | 673 | 4826 | 673 | 4826 | 673 | 4826 | 673 | 4826 |

ISNI:0000000091950263
MOFA [23] the maximum execution time was around 27 hours.

Table III reports the average CPU time taken after 20 executions by the proposed algorithms to find OGRs up to 20 marks and their comparison with average CPU time taken by GAs, and MOFA to find OGRs for optical WDM systems. The graphical representation of Table III is shown in Fig. 4. For proposed algorithms, the average CPU time varied from 0.000 second for 3-marks ruler to approximately 20 hours (for LMOFA) for 20-marks ruler. By introducing Lévy flight, hybridization of MOFA with Lévy flight and mutation. To explore the search space for MOFA, the concept of multi-population was used. The proposed algorithms have been validated and compared with other existing algorithms to find OGR sequences for WDM systems. Thus, algorithm PLMOFAM outperforms other algorithms.

V. CONCLUSIONS AND FUTURE WORK

Finding OGR sequences is an extremely challenging optimization problem. In this paper, WDM channel allocation algorithm by considering the concept of OGRs is presented. The application of improved forms of MOFA to solve OGRs problem is presented. The main technical contribution of this paper was to enhance the performance of MOFA by hybridization of MOFA with Lévy flight and mutation. To explore the search space for MOFA, the concept of multi-population was used. The proposed algorithms have been validated and compared with other existing algorithms to find
OGRs. Simulations and comparison show that the modified forms are superior to the existing algorithms. From preliminary results, it is concluded that for large order marks, algorithm PLMOFAM outperforms the other presented algorithms, as it requires less numbers of iterations and computation time to find OGRs. The outstanding performance of PLMOFAM can be very useful for the future in different multi–objective optimization design applications.

Fig. 4 Comparison of the Proposed Algorithms with Existing algorithms in Terms of Average CPU Time

REFERENCES
