International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:10, No:7, 2016

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 $(a_1 < a_2 < ... < a_n)$ such that all the positive differences $a_j - a_i, (1 \le i \le j \le n)$ are distinct [7], [8]. These non–negative integer locations are referred to as *marks* [5], [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 upto 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.

Shonak Bansal, Prince Jain, Arun Kumar Singh, and Neena Gupta are with the Electronics and Communication Engineering Department, PEC University of Technology, Chandigarh, India (e-mail: shonakk@gmail.com, princeece48@gmail.com, arun@pec.ac.in, neenagupta@pec.ac.in).

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 unpractical 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 β with distance r between any two fireflies are two main issues [32]. For a given medium having a fixed light absorption coefficient γ, the movement of a firefly i is attracted to another brighter firefly j is:

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}\left(X_j - X_i\right) + \alpha\left(rand - 0.5\right) \tag{1}$$

where I_0 is original light intensity and β_0 is attractiveness at $r = 0$. r_{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 α. For most cases in the implementation, $\beta_0 = 1$ and $\alpha \in [0,1]$.

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]:

$$f = \sum_{i=1}^{L} w_l f_l \text{ with } \sum_{i=1}^{L} w_i = 1, \qquad w_i > 0, \tag{2}$$

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 faster 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 for 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:

$$L(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \qquad (s >> s_0 > 0) \tag{3}$$

$\Gamma(\lambda)$ 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]:

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}\left(X_j - X_i\right) + \alpha . sign\left(rand - 0.5\right) \oplus L(\lambda) \tag{4}$$

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

The mutation rate probability MR_i^t 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 t is:

$$MR_i^t = \frac{f_i^t}{Max(f^t)} \tag{5}$$

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

$$F_{1i}^t = \left((LB - UB)\frac{t}{\eta} + UB\right)\beta_1 \text{ and } F_{2i}^t = \left((UB - LB)\frac{t}{\eta} + LB\right)\beta_2 \tag{6}$$

where LB, UB are lower and upper bound on the solutions respectively, β_1, $\beta_2 \in [0,1]$ are random vectors drawn from uniform distribution, and η is positive fixed parameter with large values. In order to make mutation operators F_{1i}^t and F_{2i}^t less than unity, the values of β_1, β_2 and η are selected carefully. In simplest case,

$$\beta_1 = rand_1 * 0.0001 \text{ and } \beta_2 = rand_2 * 0.0001 \tag{7}$$

$$\eta = 2 * \text{maximum number of iterations} \tag{8}$$

where $rand_1$ and $rand_2$ are random numbers between [0,1]. The mutation equation used in this paper is:

$$x_i^{t+1} = x_i^t + F_{1i}^t (x_{best}^t - x_i^t) + F_{2i}^t (x_{r1}^t - x_{r2}^t) \tag{9}$$

where x_i^t is population at iteration index t, $x_{best}^t = x_*^t$ 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),...,f_L(x),   x = (x_1,..., x_d)^T;
                                   /* d is dimension of the problem */
    Generate initial fireflies of MP populations each of size NP
    (i = 1, 2,…,MP; and j = 1, 2,…,NP);
    /* MP is multi–parallel/entire population size and NP is size of
    sub–populations in MP*/
    Define light absorption coefficient γ;
    For i = 1 : MP
      For j = 1 : NP
        Generate L weights w_l ≥ 0 so that Σ_{l=1}^{L} w_l = 1 and form a single
        objective i.e. light intensity I;
        Find the local best among ith population of NP fireflies;
      End for j
    End for i
    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 termination criterion */
      For j = 1 : MP
        For k = 1 : NP                          /*all NP fireflies*/
          For m = 1 : k
            If I_m^j > I_k^j
              Move firefly k towards m via Lévy flights;
            End if
            /* Mutation */
              Compute mutation rate probability MR;
              If (MR < rand (0,1))
                Perform mutation;
              End if
            /* End of mutation */
            Vary attractiveness with distance r via exp[− γr];
            Evaluate new generated NP solutions of jth population;
            Form single optimize objective to update light intensity;
            Rank the solutions and find current best Pareto optimal
            solution x_{lbest,m}^j ;
          End for m
        End for k
      End for j
      Find global best solution x* among the MP x_lbest solutions;
    End while
    Record x* as a non–dominated solution;
  End for i
  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 LMOFA, if both the concept of parallelism and Lévy flights are omitted the 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 $f_1(x)$ = ruler length (RL) and $f_2(x)$ = total optical channel bandwidth (TBW) are [9]:

$$RL = \sum_{i=1}^{n-1} (CS)_i \text{ subject to } (CS)_i \neq (CS)_j \tag{10}$$

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

where $i, j = 1,2,..., n$ 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), η, 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 (*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
SIMULATION PARAMETERS FOR I–MOFA'S

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

```
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 x^i_{lbest,j} among i^th population of NP fireflies by using (2), (10) and (11);
     End for j
   End for i
   Based on fitness value (Light intensity I), among MP x_lbest 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^j_m > I^j_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 x^j_{lbest,m};
           End for m
         End for k
       End for j
       Find global best solution x* among the MP x_lbest solutions;
     End while
     Record x* as a non–dominated solution;
   End for i
   Postprocess results and visualization;
End
```

Fig. 2 General Pseudo–code for I–MOFA to find OGRs for optical WDM systems

TABLE II
COMPARISON OF PROPOSED ALGORITHMS WITH EXISTING CONVENTIONAL AND NATURE–INSPIRED ALGORITHMS IN TERMS OF RULER LENGTH AND TOTAL BANDWIDTH

n	EQC [1], [4]		SA [1], [4]		GAs [9]		MOFA [22], [23]		LMOFA		MOFAM		LMOFAM		PLMOFA		PLMOFAM	
	RL	*TBW (Hz)*	*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	3	4
4	15	28	15	28	6 7	11	6 7	11	6 7	11	6 7	11	6 7	11	6 7	11	6	11
5	—	—	—	—	12 13	23 25 29	11 12 13	23 24 25	11 12 13	23 24	11 12	23 24	11 12 13	23 24	11 12 13	23 25	11 12	23 24
6	45	140	20	60	17 18 21	42 44 45	17 18	42 44	17 18	42 44	17 18	42 44	17 18	42 44	17 18	42 44	17	44
7	—	—	—	—	27 29 30	73 79 80 83	25 26 27	73 77 80 81	25 27 28	73 74 77	25 26 27	73 77 80 81	25 26 27 28	73 74 77	25 28	74 77 81 90	25 26 27	73 77
8	91	378	49	189	35 41 42	126 128 133	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117
9	—	—	—	—	52 56 59 61 63 65	192 193 196 203 225	44 49	206 208	44 49	206	44 49	206	44 47 49	185 206	44	206	44	206
10	—	—	—	—	75 76	283 287 301	55	249	55	249	55	249	55	249	55	249	55	249
11	—	—	—	—	94 96	395 456	72	391	72	386	72 103	378 391	72 103	378 386 391	72	386	72	386
12	231	1441	132	682	123 128 137	532 581 660	85	515	85	503	85	503	85	503	85	503	85	503
13	—	—	—	—	203 241	1015 1048	106	725 744	106 111	675 725	106 111	673 720	106	660	106	660	106	660
14	325	2340	286	1820	206 228 230	1172 1177 1285	169 206	991 1001	206	991	169	1001	127	924	127	924	127	924
15	—	—	—	—	275 298	1634 1653	260	1554	151	1047	151	1047	151	1047	151	1047	151	1047
16	—	—	—	—	316	1985	283	1804	283	1804	283	1804	177	1298	177	1298	177	1298
17	—	—	—	—	355	2205	355	2205	354	2208	354	2208	369	2201	199	1661	199	1661
18	561	5203	493	5100	427 463	2599 3079	463	2599	362	2912	445	2566	445 427	2566 3079	216	1894	216	1894
19	—	—	—	—	567 597	3432 5067	567	3432	467	3337	475 467	3408 3337	467	3337	246	2225	246	2225
20	703	7163	703	6460	615 673 680 691	4660 4826 4905 4941	649	4517	615	4660	615	4660	578	4306	283	2794	283	2794

C. Comparison of Proposed Algorithms in Terms of Computational Time

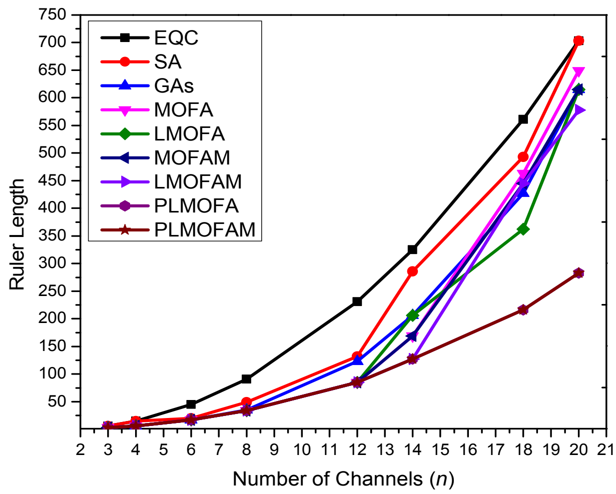
Finding OGRs for higher order marks by exhaustive search algorithms are very time consuming, which means that it takes several hours, months, and even years of calculation on the network of several thousand computers [6], [7], [34], [35]. In [17], it is identified that to find Golomb ruler sequences from heuristic based exhaustive search algorithm, the times varied from 0.035 seconds to 6 weeks for 5 to 13–marks ruler, whereas by non–heuristic exhaustive search algorithms took approximately 12.57 minutes for 10–marks, 2.28 years for 12–marks, 2.07e+04 years for 14–marks, 3.92e+09 years for 16–marks, 1.61e+15 years for 18–marks and 9.36e+20 years for 20–marks ruler. 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

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

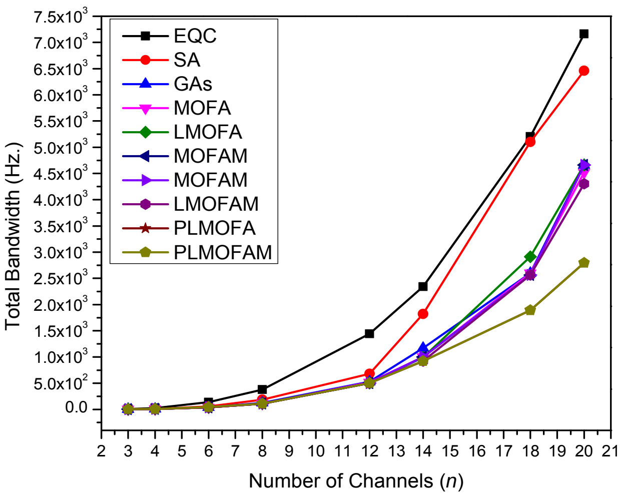
TABLE III
COMPARISON OF AVERAGE CPU TIME TAKEN BY PROPOSED ALGORITHMS WITH GAS, AND MOFA

n	**GAs [9]** *Average CPU time (Sec.)*	**MOFA [23]** *Average CPU time (Sec.)*	**LMOFA** *Average CPU time (Sec.)*	**MOFAM** *Average CPU time (Sec.)*	**LMOFAM** *Average CPU time (Sec.)*	**PLMOFA** *Average CPU time (Sec.)*	**PLMOFAM** *Average CPU time (Sec.)*
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.001	0.000	0.000	0.000	0.000	0.000	0.000
5	0.021	0.011	0.001	0.001	0.001	0.001	0.000
6	0.780	0.4398	0.1238	0.1088	0.1013	0.0211	0.0201
7	1.120	0.8520	0.4899	0.3895	0.1870	0.0478	0.0479
8	1.241	1.0227	0.7441	0.7321	0.2379	0.0693	0.0601
9	1.711	1.4890	1.9872	1.5680	1.3750	0.0886	0.0798
10	5.499e+01	5.211e+01	3.149e+01	3.138e+01	3.111e+01	0.5271	0.4108
11	7.200e+02	6.710e+02	4.766e+02	4.767e+02	4.645e+02	1.6976	1.4332
12	8.602e+02	7.890e+02	5.658e+02	5.652e+02	5.648e+02	9.456	4.8891
13	1.070e+03	1.010e+03	8.750e+02	8.751e+02	8.436e+02	3.451e+01	3.224e+01
14	1.028e+03	1.019e+03	1.014e+03	1.012e+03	0.981e+03	5.529e+01	4.993e+01
15	1.440e+03	1.270e+03	1.166e+03	1.165e+03	1.090e+03	8.334e+01	7.987e+01
16	1.680e+03	1.439e+03	1.342e+03	1.342e+03	1.158e+03	4.881e+02	3.778e+02
17	5.048e+04	4.041e+03	3.460e+03	3.455e+03	3.320e+03	6.678e+02	5.899e+02
18	6.840e+04	5.875e+04	4.075e+04	4.076e+04	3.880e+04	9.997e+02	8.975e+02
19	8.280e+04	7.132e+04	6.687e+04	6.688e+04	6.390e+04	4.897e+03	3.597e+03
20	1.12428e+05	9.876e+04	7.335e+04	7.432e+04	7.110e+04	8.022e+03	7.889e+03

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, mutation strategies and multiple populations with MOFA, average CPU time is reduced to approximately 2.2 hours for PLMOFAM. This represents the improvement achieved by the modified forms of MOFA to find OGR sequences for WDM systems. Thus, algorithm PLMOFAM outperforms other algorithms.



(a)



(b)

Fig. 3 Comparison of the Proposed Algorithms with Existing algorithms in Terms of (a) Ruler Length, and (b) Total Bandwidth

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.

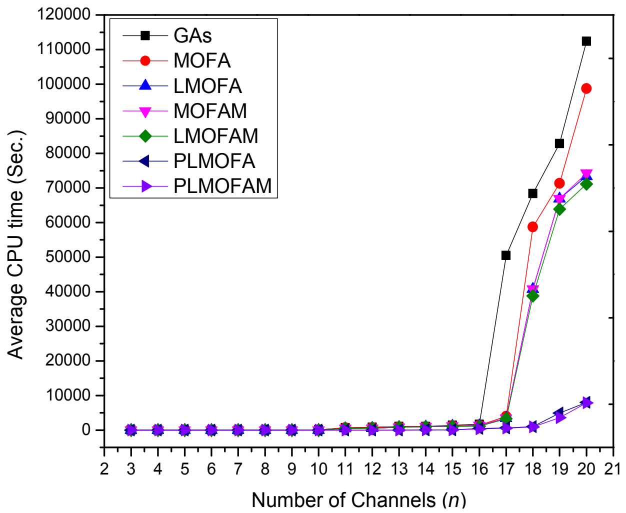


Fig. 4 Comparison of the Proposed Algorithms with Existing algorithms in Terms of Average CPU Time

REFERENCES

[1] W. C. Kwong, and G. C. Yang, “An Algebraic Approach to the Unequal–Spaced Channel–Allocation Problem in WDM Lightwave Systems”, IEEE Transactions on Communications, Vol. 45, No. 3, pp. 352–359, March–1997.
[2] V. L. L. Thing, P. Shum, and M. K. Rao, “Bandwidth–Efficient WDM Channel Allocation for Four–Wave Mixing–Effect Minimization”, IEEE Transactions on Communications, Vol. 52, No. 12, pp. 2184–2189, December 2004.
[3] O. K. Tonguz and B. Hwang, “A Generalized Suboptimum Unequally Spaced Channel Allocation Technique—Part II: In coherent WDM systems”, IEEE Transactions on Communications, Vol. 46, pp. 1186–1193, September–1998.
[4] R. Randhawa, J. S. Sohal, R. S. Kaler, “Optimum Algorithm for WDM Channel Allocation for Reducing Four–Wave Mixing Effects”, Optik 120 (2009), pp. 898–904, 2009.
[5] G. S. Bloom and S. W. Golomb, “Applications of Numbered Undirected Graphs”, Proceedings of the IEEE, Vol. 65, No. 4, pp. 562–570, April–1977.
[6] J. B. Shearer, “Some New Disjoint Golomb Rulers”, IEEE Transactions on Information Theory, Vol. 44, No. 7, pp. 3151–3153, November–1998.
[7] Distributed.net, “Project OGR”. Available at http://www.distributed.net/ogr.
[8] K. Drakakis and S. Rickard, “On the Construction of Nearly Optimal Golomb Rulers by Unwrapping Costas Arrays”, Contemporary Engineering Sciences, Vol. 3, No. 7, pp. 295–309, July–2010.
[9] S. Bansal, “Optimal Golomb Ruler Sequence Generation for FWM Crosstalk Elimination: Soft Computing Versus Conventional Approaches”, Applied Soft Computing, Vol. 22, pp. 443–457, September–2014.
[10] http://mathworld.wolfram.com/PerfectRuler.html.
[11] http://mathworld.wolfram.com/GolombRuler.html.
[12] C. Meyer and P. A. Papakonstantinou, “On the complexity of constructing Golomb Rulers”, Discrete Applied Mathematics, Vol. 157, pp. 738–748, 2009.
[13] N. Memarsadegh, “Golomb Patterns: Introduction, Applications, and Citizen Science Game”, Information Science and Technology (IS&T), Seminar Series NASA GSFC, September 11, 2013. Available at http://istcolloq.gsfc.nasa.gov/fall2013/presentations/memarsadeghi.pdf.
[14] J. P. Robinson, “Optimum Golomb Rulers”, IEEE Transactions on Computers, Vol. 28, No. 12, pp. 183–184, December–1979.
[15] J. B. Shearer, “Some New Optimum Golomb Rulers”, IEEE Transactions on Information Theory, Vol. 36, pp. 183–184, January–1990.
[16] C. Cotta, I. Dotú, A. J. Fernández, and P. V. Hentenryck, “A Memetic Approach to Golomb Rulers”, Parallel Problem Solving from Nature–PPSN IX, Lecture Notes in Computer Science, Vol. 4193, pp. 252–261, Springer–Verlag Berlin Heidelberg, September 9–13, 2006.
[17] S. W. Soliday, A. Homaifar and Gary L. Lebby, “Genetic Algorithm Approach to the Search for Golomb Rulers”, Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA–95), Morgan Kaufmann, pp. 528–535, 1995.
[18] J. P. Robinson, “Genetic Search for Golomb Arrays”, IEEE Transactions on Information Theory, Vol. 46, No. 3, pp. 1170–1173, May–2000.
[19] I. Dotú, P. V. Hentenryck, “A Simple Hybrid Evolutionary Algorithm for Finding Golomb Rulers”, Evolutionary Computation, 2005, The 2005 IEEE Congress on, Vol. 3, pp. 2018–2023, September 2–5, 2005.
[20] N. Ayari; Thé Van Luong and A. Jemai, “A Hybrid Genetic Algorithm for Golomb Ruler Problem”, ACS/IEEE International Conference on Computer Systems and Applications (AICCSA–2010), pp.1–4, May 16–19, 2010.
[21] S. Bansal, S. Kumar and P. Bhalla, “A Novel Approach to WDM Channel Allocation: Big Bang–Big Crunch Optimization”, In the Proceeding of Zonal Seminar on Emerging Trends in Embedded System Technologies (ETECH–2013) organized by The Institution of Electronics and Telecommunication Engineers (IETE)”, Chandigarh Centre, Chandigarh, pp. 80–81, 2013.
[22] S. Bansal and K. Singh, “A Novel Soft–Computing Algorithm for Channel Allocation in WDM Systems”, International Journal of Computer Applications (IJCA), Vol. 85, No. 9, pp. 19–26, January–2014.
[23] K. Singh, “Soft Computing Algorithms fot WDM Channel Allocation for Reducing FWM Crosstalk”, M.Tech. Thesis, Department of Electronics and Communication Engineering, Institute of Science and Technology, Klawad, India, December–2013.
[24] S. Bansal, R. Chauhan and P. Kumar, “A Cuckoo Search based WDM Channel Allocation Algorithm”, International Journal of Computer Applications (IJCA), Vol. 96, No. 20, pp. 6–12, June–2014.
[25] S. Bali, S. Bansal and A. Kamboj, “A Novel Hybrid Multi–objective BB–BC Based Channel Allocation Algorithm to Reduce FWM Crosstalk and its Comparative Study”, International Journal of Computer Applications (IJCA), Vol. 124, No. 12, pp. 38–45, August–2015.
[26] P. Jain, S. Bansal, A. K. Singh, and N. Gupta, “Golomb Ruler Sequences Optimization for FWM Crosstalk Reduction: Multi–population Hybrid Flower Pollination Algorithm”, Progress in Electromagnetics Research Symposium (PIERS), Prague, Czech Republic, pp. 2463–2467, July 06–09, 2015.
[27] S. Bali, S. Bansal, and A. Kamboj, “A Novel Hybrid Multi–objective BB–BC Based Channel Allocation Algorithm to Reduce FWM Crosstalk and its Comparative Study”, International Journal of Computer Applications (IJCA), Vol. 124, No. 12, pp. 38–45, August–2015.
[28] X.–S. Yang, “Flower Pollination Algorithm for Global Optimization”, in: Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, Vol. 7445, Springer, Berlin, pp. 240–249, 2012.
[29] R. Storn and K. V. Price, “Differential Evolution—A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces,” Journal of Global Optimization, Vol. 11, No. 4, pp. 341–359, December–1997.
[30] S. Koziel and X.–S. Yang, “Computational Optimization, Methods and Algorithms”, Studies in Computational Intelligence, Vol. 356, Springer, 2011.
[31] X.–S. Yang, “Multiobjective Firefly Algorithm for Continuous Optimization”, Engineering with Computers, Vol. 29, No. 2, pp. 175–184, 2013.
[32] X.–S. Yang, “Firefly Algorithms for Multimodal Optimization”, in: Stochastic Algorithms: Foundations and Applications (SAGA–2009), Lecture Notes in Computer Science, Vol. 5792, Springer–Verlag, Berlin, pp. 169–178, 2009.

[33] X.–S. Yang, "Review of Metaheuristics and Generalized Evolutionary Walk Algorithm", International Journal of Bio–Inspired Computation (IJBIC), Vol. 3, No. 2, pp. 77–84, 2011.
[34] A. Dollas, W. T. Rankin, and D. McCracken, "A New Algorithm for Golomb Ruler Derivation and Proof of the 19 Mark Ruler", IEEE Transactions on Information Theory, Vol. 44, No. 1, pp. 379–382, January–1998.
[35] J. B. Shearer, "Golomb Ruler Table", Mathematics Department, IBM Research. Available at http://www.research.ibm.com/people/s/shearer/grtab.html, 2001.