Artificial Intelligence Review (2020) 53:5589–5635
https://doi.org/10.1007/s10462-020-09829-2

Performance comparison of five metaheuristic nature-inspired algorithms to find near-OGRs for WDM systems

Shonak Bansal[1]

Published online: 8 April 2020

Abstract
The metaheuristic approaches inspired by the nature are becoming powerful optimizing algorithms for solving NP-complete problems. This paper presents five nature-inspired metaheuristic optimization algorithms to find near-optimal Golomb ruler (OGR) sequences in a reasonable time. In order to improve the search space and further improve the convergence speed and optimization precision of the metaheuristic algorithms, the improved algorithms based on mutation strategy and Lévy-flight search distribution are proposed. These two strategies help the metaheuristic algorithms to jump out of the local optimum, improve the global search ability so as to maintain the good population diversity. The OGRs found their potential application in channel-allocation method to suppress the four-wave mixing crosstalk in optical wavelength division multiplexing systems. The results conclude that the proposed algorithms are superior to the existing conventional computing algorithms i.e. extended quadratic congruence and search algorithm and nature-inspired optimization algorithms i.e. genetic algorithms, biogeography based optimization and simple big bang–big crunch to find near-OGRs in terms of ruler length, total optical channel bandwidth and computation time. The idea of computational complexity for the proposed algorithms is represented through the Big O notation. In order to validate the proposed algorithms, the non-parametric statistical Wilcoxon analysis is being considered.

Keywords Channel spacing · Conventional computing · Equally and unequally spaced channel allocation · Four-wave mixing · Metaheuristic · Nature-inspired algorithm · Near-optimal Golomb ruler · Optimization

1 Introduction

There exists a rich collection of nonlinear optical effects (Kwong and Yang 1997; Aggarwal 2001; Thing et al. 2004; Babcock 1953; Singh and Bansal 2013) in optical WDM systems, each of which manifests itself in a unique way. Out of these nonlinearities, the FWM

✉ Shonak Bansal
shonakk@gmail.com

[1] Electronics and Communication Engineering Department, Punjab Engineering College (Deemed to be University), Sector-12, Chandigarh, India

crosstalk signal is the major dominant noise effects in optical WDM systems employing equal channel spacing (ECS). Four-wave mixing is a third-order nonlinear optical effect in which two or more wavelengths (or frequencies) combine and produce several mixing products. For uniformly spaced WDM channels, the generated FWM product terms fall onto other active channels in the band, causing inter-channel crosstalk. The performance can be substantially improved if FWM crosstalk generation at the channel frequencies is prevented. The efficiency of FWM signals depends on the channel spacing and fiber dispersion. If the frequency separation of any two channels of an optical WDM system is different from that of any other pair of channels, no FWM crosstalk signals will be generated at any of the channel frequencies (Kwong and Yang 1997; Aggarwal 2001; Thing et al. 2004; Babcock 1953; Singh and Bansal 2013).

To suppress the FWM signals in optical WDM systems, unequally spaced channel allocation (USCA) algorithms (Kwong and Yang 1997; Sardesai 1999; Forghieri et al. 1995; Hwang and Tonguz 1998; Tonguz and Hwang 1998; Atkinson et al. 1986; Randhawa et al. 2009) have been proposed, having the limitation of increased channel bandwidth requirement compared to equally spaced channel allocation (ESCA). This paper proposes an unequally spaced optical bandwidth channel allocation algorithm by taking into consideration the concept of near-OGRs (Babcock 1953; Bloom and Golomb 1977; Shearer 1998) to suppress FWM crosstalk in optical WDM systems.

Studies have been shown that Golomb rulers represent a class of NP-complete (http://theinf1.informatik.uni-jena.de/teaching/ss10/oberseminar-ss10) problems. For higher order marks, the exhaustive computer search (Robinson 1979; Shearer 1990) of such problems is difficult. Numerous algorithms (Robinson 1979; Shearer 1990; Galinier et al. 2001; Leitao 2004; Rankin 1993; Cotta et al. 2006) have been proposed to tackle Golomb ruler problem. To date, no efficient algorithm is known for finding the shortest length ruler. The realization of nature-inspired metaheuristic optimization algorithms such as Tabu search (TS) (Cotta et al. 2006), Memetic approach (Cotta et al. 2006), Genetic algorithms (GAs) (Soliday et al. 1995; Robinson 2000; Ayari et al. 2010; Dotú and Hentenryck 2005) and its hybridizations (HGA) (Ayari et al. 2010), hybrid evolutionary (HE) algorithms (Dotú and Hentenryck 2005), Biogeography based optimization (BBO) (Bansal 2014) and Big bang–big crunch (BB–BC) (Bansal 2017; Bansal and Sharma 2017) in finding relatively good solutions to such NP-complete problems provides a good starting point for algorithms of finding near-OGRs. Therefore, nature-inspired algorithms seem to be very effective solutions for such NP-complete problems. This paper proposes the application of five nature-inspired algorithms namely BB–BC algorithm, Firefly algorithm (FA), Bat algorithm (BA), Cuckoo search algorithm (CSA), Flower pollination algorithm (FPA) and their modified forms to find either optimal or near-optimal rulers in a reasonable time and their performance comparison with the existing conventional and nature-inspired algorithms to find near-OGRs.

2 Golomb rulers

Babcock (1953) firstly introduced the concept of *Golomb rulers*, and further was described by Bloom and Golomb (1977). According to the literatures (Colannino 2003; Dimitromanolakis 2002; Dollas et al. 1998), all of rulers' up to 8-marks introduced in Babcock (1953) are optimal; the 9- and 10-marks are near-optimal.

Golomb rulers are an ordered set of unevenly marks at positive integer locations such that no distinct pairs of numbers from the set have the same difference (http://www.distributed.net/ogr; Bansal 2019; Drakakis and Rickard 2010; Drakakis 2009). This means that an n-mark Golomb ruler $G = \{x_1, x_2, \ldots, x_{n-1}, x_n\}$ $x_1 < x_2 < \cdots < x_{n-1} < x_n$ is an ordered set of n different positive integer numbers such that all the positive differences

$$|x_i - x_j|, \quad x_i, x_j \in G \quad \forall\, i > j \,\text{or}\, i \neq j \tag{1}$$

are distinct (Bansal 2017). And

$$\forall i, j, k, l \in \{1, 2, \ldots, n-1, n\}, x_i - x_j = x_k - x_l \Leftrightarrow i = k \wedge j = l. \tag{2}$$

The positive integer numbers are referred to as *order* or *marks*. The number of marks on a ruler is referred to as the *ruler size*. The difference between the largest and smallest number is referred to as the *ruler length RL*, i.e.

$$RL = \max(G) - \min(G) = x_n - x_1 \tag{3}$$

where

$$\max(G) = \max\{x_1, x_2, \ldots, x_{n-1}, x_n\} = x_n \tag{4}$$

and

$$\min(G) = \min\{x_1, x_2, \ldots, x_{n-1}, x_n\} = x_1 \tag{5}$$

Generally, the first mark x_1 of set G may be assumed on position 0. Then the n-mark Golomb ruler set becomes $G = \{0, x_2, \ldots, x_{n-1}, x_n\}$ and the RL of such n-mark set G is x_n.

An OGR is the shortest length ruler for a given number of marks (http://mathworld.wolfram.com/PerfectRuler.html; http://mathworld.wolfram.com/GolombRuler.html). There can be multiple different OGRs for a specific number of marks. However, the unique optimal Golomb 4-marks ruler is shown in Fig. 1, which measures all the distances from 0 to 6.

A perfect Golomb ruler measures all the integer distances from 0 to RL (http://theinf1.informatik.uni-jena.de/teaching/ss10/oberseminar-ss10; Soliday et al. 1995). The ruler length RL of perfect Golomb ruler set G is (Rankin 1993):

$$RL = \frac{n(n-1)}{2} = \sum_{i=1}^{n-1} i \tag{6}$$

For example, the set (0, 1, 3, 7, 12, 20), shown in Fig. 2 is a non-optimal 6-marks Golomb ruler with a length of 20. As from the differences it is clear that the numbers 10, 14, 15, 16, 18 are missing, so it is not a perfect Golomb ruler set. The distance associated between each pair of marks is also shown in Fig. 2.

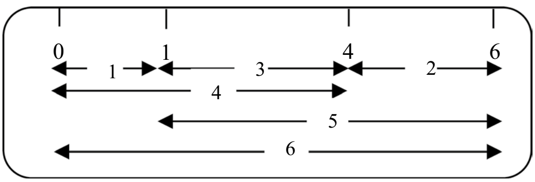


Fig. 1 A 4-marks OGR with its associated distances

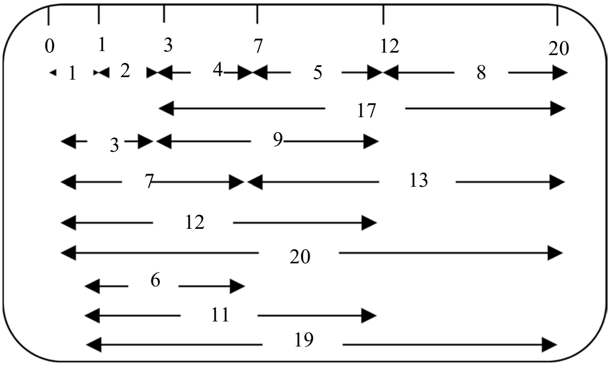


Fig. 2 A 6-marks non-OGR with its associated distances

The OGRs found their potential applications in radio frequency allocation, computer communication network, sensor placement in X-ray crystallography, circuit layout, pulse phase modulation, self-orthogonal codes, VLSI architecture, geographical mapping, coding theory, linear arrays, fitness landscape analysis, radio astronomy, antenna design for radar missions, sonar applications and NASA missions in astrophysics, planetary and earth sciences (Babcock 1953; Bloom and Golomb 1977; Rankin 1993; Soliday et al. 1995; Dimitromanolakis 2002; Dollas et al. 1998; Lam and Sarwate 1988; Lavoie et al. 1991; Robinson and Bernstein 1967; Cotta and Fernández 2005; Fang and Sandrin 1977; Blum et al. 1974; Memarsadegh 2013; http://encompass.gsfc.nasa.gov/cases.html).

On applying OGRs to the channel allocation, it was possible to achieve the smallest distinct number to be used for the optical WDM channel allocation problem. As the difference between any two numbers is distinct, the new FWM frequency signals generated would not fall into the one already assigned for the carrier channels.

3 Nature-inspired metaheuristic algorithms

Due to highly nonlinearity and complexity of the problem of interest, design optimization in engineering fields tends to be very challenging. As conventional computing algorithms are local search algorithm, so they are not the best tools for highly nonlinear global optimization, and thus often miss the global optimality. In addition, design solutions have to be robust, low cost, subject to uncertainty in parameters and tolerance for imprecision of available components and materials. Nature-inspired algorithms are now among the most widely used optimization algorithms. The guiding principle is to devise algorithms of computation that lead to an acceptable solution at low cost by seeking for an approximate solution to a precisely/imprecisely formulated problem (Cotta and Hemert 2008; Yang 2010a, 2012a, 2013a; Koziel and Yang 2011; Rajasekaran and Vijayalakshmi Pai 2004; Mitchell 2004).

This section is devoted to the brief overview of nature-inspired optimization algorithms based on the theories of big bang and big crunch called BB–BC, flash pattern of fireflies called FA, the echolocation characteristics of microbats called BA, brood parasitism of cuckoo species called CSA and flow pollination process of flowering plants called FPA.

The power of nature-inspired optimization algorithms lies in how faster the algorithms explore the new possible solutions and how efficiently they exploit the solutions to make them better. Although all optimization algorithms in their simplified form works well in the exploitation (the fine search around a local optimal), there are some problems in the global exploration of the search space. If all of the solutions in the initial phase of the optimization algorithm are collected in a small part of search space, the algorithms 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 the optimization algorithms, there is a need by which exploration and exploitation can be enhanced and the algorithms can work more efficiently. By keeping this in mind two features, fitness (cost) based mutation strategy and random walk i.e. Lévy-flight distribution are introduced in the proposed metaheuristic algorithms, which is the main technical contribution of this paper. The Lévy flights distribution is much faster than the normal random walk. Lévy flights can reduce the required number of metaheuristic algorithms iterations by ~4 orders compared to normal random walk (Mareli and Twala 2018). Both the mutation and Lévy flight strategies help the metaheuristic algorithms to jump out of the local optimum, avoid the premature convergence of the algorithm and improve the global search ability so as to maintain the good population diversity. In all the modified algorithms, the mutation rate probability is determined based on the fitness value. The mutation rate probability MR_i^t of each solution x_i at running iteration index t, mathematically is given by:

$$MR_i^t = \frac{f_i^t}{Max(f^t)} \tag{7}$$

where f_i^t is the fitness value of each solution x_i at iteration index t, and $Max(f^t)$ is the maximum fitness value in the population at iteration t. For all proposed algorithms, the mutation equation (Storn and Price 1997; Price et al. 2005) use throughout this paper is:

$$x_i^t = x_i^{t-1} + p_m(x_{best}^{t-1} - x_i^{t-1}) + p_m(x_{r_1}^{t-1} - x_{r_2}^{t-1}) \tag{8}$$

where x_i^t is the population at running iteration index t, $x_{best}^{t-1} = x_*^{t-1}$ is the current global best solution at iteration one less than running iteration index t, p_m is mutation operator, r_1 and r_2 are uniformly distributed random integer numbers between 1 to size of the given problem. The numbers r_1 and r_2 are different from running index. Typical values of p_m are same as in GA i.e. 0.001 to 0.05. The mutation strategy increases the chances for a good solution, but a high mutation rate (p_m=0.5 and 1.0) results in too much exploration and is disadvantageous to the improvement of candidate solutions. As p_m decreases from 1.0 to 0.01, optimization ability increases greatly, but as p_m continues to decrease to 0.001, optimization ability decreases rapidly. A small value of p_m is not able to sufficiently increase solution diversity (Bansal 2014).

The Lévy flight distribution (Yang 2012b) used for all proposed algorithms in this paper mathematically is given by:

$$L(\lambda) \sim \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s >> s_0 > 0) \tag{9}$$

Here, $\Gamma(\lambda)$ is the standard gamma distribution valid for large steps i.e. for $s>0$. Throughout the paper, $\lambda = 3/2$ is used. In theory, it is required that $|s_0| >> 0$, but in practice s_0 can be as small as 0.1 (Yang 2012b).

By introducing these two features in the simplified forms of proposed algorithms, the basic concept of search space is modified i.e. the proposed algorithms can explore new search space by the mutation and random walk. A fundamental benefit of using mutation and Lévy flight strategies with nature-inspired algorithms in this paper is their ability to improve its solutions over time, which does not seem in the existing algorithms (Cotta et al. 2006; Soliday et al. 1995; Robinson 2000; Ayari et al. 2010; Dotú and Hentenryck 2005; Bansal 2014, 2017; Bansal and Sharma 2017) to find near-OGRs.

3.1 Big bang–big crunch optimization algorithm and its modified forms

Erol and Eksin (2006), inspired by the theories of the evolution of universe; namely, the Big bang and Big crunch theory, developed a metaheuristic algorithm called Big bang–big crunch (BB–BC) optimization algorithm. BB–BC algorithm has two phases: Big bang phase where candidate solutions are randomly distributed over the search space and big crunch phase where a contraction procedure calculates a center of mass or the best fit individual for the population (Erol and Eksin 2006; Afshar and Motaei 2011; Tabakov 2011; Yesil and Urbas 2010). In BB–BC, the centre of mass mathematically is computed by:

$$x_c = \frac{\sum_{i=1}^{Popsize} \frac{1}{f_i} x_i}{\sum_{i=1}^{Popsize} \frac{1}{f_i}} \tag{10}$$

where x_c = position of the center of mass; x_i = position of candidate i; f_i = fitness (cost) value of candidate i; and $Popsize$ = population size. Instead of the center of mass, the best fit individual can also be chosen as the starting point in the big bang phase. The new candidates (x_{new}) around the centre of mass are calculated by adding or subtracting a normal random number whose value decreases as the iterations elapse. This can be formalized as by (Erol and Eksin 2006):

$$x_{new} = x_c + r \times c_1 \times \frac{(x_{\max} - x_{\min})}{1 + t/c_2} \tag{11}$$

where r is a random number with a standard normal distribution, c_1 is a parameter for limiting the size of the search space, parameter c_2 denotes after how many iterations the search space will be restricted to half, x_{max} and x_{min} are the upper and lower limits of elite pool, and t is the iteration index.

If fitness based mutation strategy is introduced in the simple BB–BC algorithm, a new *Big bang–big crunch algorithm with mutation* (BB–BCM) can be formulated.

On adding Lévy-flight distributions in the simple BB–BC algorithm, another new *Lévy–flight Big bang–big crunch algorithm* (LBB–BC) can be formulated. For LBB–BC, Eq. (11) is randomized via Lévy flights as:

$$x_{new} = x_c + r \times c_1 \times \frac{(x_{\max} - x_{\min})}{1 + t/c_2} \oplus L(\lambda) \tag{12}$$

The product $\oplus$ means entrywise multiplications and $L(\lambda)$ is the Lévy flight based step size given mathematically by Eq. (9).

If fitness based mutation strategy is applied to LBB–BC algorithm, *Lévy flight Big bang–big crunch with mutation* (LBB–BCM) algorithm can be formulated.

Based upon the above discussions, the corresponding general pseudo-code for modified BB–BC algorithm (MBB–BC) can be summarized in Fig. 3. If the lines 17–22 in Fig. 3 are removed and Lévy flight distributions in lines 14–16 are not used, then Fig. 3 represents the general pseudo-code for BB–BC algorithm. If from lines 14–16 Lévy flight distributions are not used, then Fig. 3 corresponds to the general pseudo-code for BB–BCM algorithm. If no modifications in Fig. 3 are performed, then it represents the pseudo-code for LBB–BCM algorithm.

3.2 Firefly algorithm and its modified forms

Yang (2010a, 2012a, 2013a), Koziel and Yang (2011) inspired by the flashing pattern and characteristics of fireflies, developed a novel optimization algorithm called Firefly inspired algorithm or Firefly algorithm (FA). For describing this algorithm, FA uses the following three idealized rules:

1. All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex;
2. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly;
3. The brightness of a firefly is determined by the landscape of the objective function.

In FA, the variation of light intensity and the formulation of attractiveness are two main issues. For maximum optimization problems, the brightness I of a firefly at a particular

Fig. 3 General pseudo-code for MBB–BC Algorithm

```
1. Modified Big Bang–Big Crunch (MBB–BC) Algorithm
2. Begin
3.   /* Big Bang Phase */
4.        Generate a random set of candidates (population);
5.   /* End of Big Bang Phase */
6.   While not t                          /* t is a termination criterion */
7.     Compute the fitness value of all the candidate soluti ons;
8.     Sort the population from best to worst based on fitness (cost)
9.     value;
10.    /* Big Crunch Phase */
12.        Compute the center of mass;
13.    /* End of Big Crunch Phase */
14.    Calculate new candidates around the center of mass by adding
15.    or subtracting a normal random number whose value decreases
16.    as the iterations elapse via Lévy flight;      /* Big Bang Phase */
17.    /* Mutation */
18.        Compute mutation rate probability MR_i via equation (7);
19.        If (MR_i < rand)
20.          Perform mutation via equation (8);
21.        End if
22.    /* End of mutation */
23.    Rank the candidates and find the current best;
24.  End while
25.  Postprocess results and visualization;
26. End
```

location X can simply be proportional to the objective function i.e. $I(X) \propto f(X)$ (Yang 2010a, b, c, 2011a, b, 2012a, 2013a; Koziel and Yang 2011; Yang and Deb 2010a; Yang and He 2013). As both the light intensity and attractiveness decreases as the distance from the source increases, the variations of the light intensity and attractiveness should be monotonically decreasing functions. For a given medium with a fixed light absorption coefficient γ, the light intensity $I(r)$ varies with the distance r between any two fireflies (Yang 2010b) as:

$$I = I_0 e^{-\gamma r} \tag{13}$$

where I_0 denotes the original light intensity.

As attractiveness of a firefly is proportional to the light intensity seen by the neighboring fireflies, therefore the attractiveness β of a firefly with the distance r is given by:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{14}$$

where β_0 is the attractiveness at $r = 0$.

The distance between any two fireflies i and j at locations X_i and X_j, respectively, is the Cartesian distance as given by (Yang 2010b):

$$r_{ij} = \left\| X_i - X_j \right\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k} \right)^2} \tag{15}$$

where $x_{i,k}$ is the kth component of the spatial coordinate X_i of i^{th} firefly and d is the number of dimensions in search space. The movement of a firefly i is attracted to another brighter firefly j is determined by (Yang 2010b):

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2} \left(X_j - X_i \right) + \alpha(rand - 0.5) \tag{16}$$

where the second term is due to the attraction and the third term is randomization with a control parameter α, which makes the more efficient exploration of the search space. For most cases in the implementation,$\beta_0 = 1$ and $\alpha \in [0, 1]$.

If mutation strategy is combined with the above mentioned three idealized rules, *Firefly algorithm with mutation* (FAM) can be formulated. All the parameters and equations for FAM are same as for simple FA. Only the difference between algorithms FAM and simple FA is that mutation strategy is added to simple FA.

By combining the characteristics of Lévy flights with the simple FA, another new algorithm named, *Lévy flight Firefly algorithm* (LFA) can be formulated. For LFA, the third term in Eq. (16) is randomized via Lévy flights. The firefly movement equation for LFA is approximated by:

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2} \left(X_j - X_i \right) + \alpha.sign(rand - 0.5) \oplus L(\lambda) \tag{17}$$

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 having an infinite variance with an infinite mean. In LFA the steps of firefly motion are essentially a random walk process.

If both algorithms FAM and LFA are combine into a single algorithm, then *Lévy flight Firefly algorithm with mutation* (LFAM) can be formulated.

The corresponding general pseudo-code for modified FA (MFA) is shown in Fig. 4. If lines 15–20 in Fig. 4 are removed and in line 13 Lévy flight distributions are not used, then

Fig. 4 General pseudo-code for MFA

```
1. Modified Firefly Algorithm (MFA)
2.  Begin
3.   /* MFA parameter initialization */
4.        Define objective function f (X);
5.        Generate initial population of fireflies x_i (i = 1, 2,…n);
6.        Compute the light intensity I_i at x_i by f (X_i);
7.        Define light absorption coefficient γ;
8.   /* End of MFA parameter initialization */
9.   While not t                    /* t is a termination criterion */
10.    For i = 1 : n                              /*all n fireflies*/
11.      For j = 1 : i
12.        If (I_j > I_i)
13.          Move firefly i towards j in d-dimension via Lévy flights;
14.        End if
15.        /* Mutation */
16.            Compute mutation rate probability MR_i via equation (7);
17.            If (MR_i < rand)
18.              Perform mutation via equation (8);
19.            End if
20.        /* End of mutation */
21.        Vary attractiveness with distance r via exp[− γr];
22.        Evaluate new solutions and update light intensity;
23.      End for                                     /* End for j */
24.    End for                                       /* End for i */
25.    Rank the fireflies and find the current best;
26.  End while
27.  Postprocess results and visualization;
28. End
```

Fig. 4 corresponds to the general pseudo-code for simple FA. If Lévy flight distributions in line 13 are not used in Fig. 4, then it corresponds to the general pseudo-code for FAM and if no modifications in Fig. 4 are performed then it represents the general pseudo-code for LFAM algorithm.

3.3 Bat algorithm and its modified forms

Yang (2010a, c, 2011b, 2012a, Yang 2013a) and Koziel and Yang (2011), inspired by the echolocation characteristics of microbats, introduced a novel optimization algorithm called Bat algorithm (BA). For describing this new algorithm, the author in Yang (2010c) uses the following three idealized rules:

1. To sense the distance, all bats use echolocation and they also *know* the surroundings in some magical way;
2. Bats fly randomly with velocity v_i at position x_i, with a fixed frequency range $[f_{min}, f_{max}]$, fixed wavelength range $[\lambda_{min}, \lambda_{max}]$, varying its pulse emission rate $r \in [0, 1]$, and loudness A_0 to hunt for prey, depending on the proximity of their target;
3. Although the loudness can vary in different ways, it is assume that the loudness varies from a minimum constant (positive) A_{min} to a large A^0.

In BA, each bat is defined by its position x_i, velocity v_i, frequency f_i, loudness A_i, and the emission pulse rate r_i in a d-dimensional search space. Among all the bats, there is a current global best solution x_* which is located after comparing all the solutions among all the bats. The new velocities v_i^t and solutions x_i^t at step t are given by (Yang 2010c, 2013a, b; Li et al. 2019; Guo et al. 2019):

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{18}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*) f_i \tag{19}$$

$$x_i^t = x_i^{t-1} + v_i^{t-1} \tag{20}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. A random walk is used for local search that modifies the current best solution according to Eq. (21):

$$x_{new} = x_{best} + \varepsilon A^t \tag{21}$$

where $x_{best} = x_*$, $\varepsilon \in [-1, 1]$ is a scaling factor and A^t is loudness. Further the loudness A and pulse rate r are updated according to the Eqs. (22) and (23) respectively as iterations proceed:

$$A_i^t = \alpha A_i^{t-1} \tag{22}$$

$$r_i^t = r_i^0 [1 - e^{-\gamma t}] \tag{23}$$

where α and γ are constants and for simplicity, $\alpha = \gamma$ is chosen. For most of the simulation $\alpha = \gamma = 0.9$ is used (Yang 2010c).

By combining the characteristics of mutation and Lévy flights strategies with the simple BA, three new algorithms, namely, *Bat algorithm with mutation* (BAM), *Lévy flight Bat algorithm* (LBA) and *Lévy flight Bat algorithm with mutation* (LBAM) can be formulated. For LBA, the modification performed in Eq. (21) is given by:

$$x_{new} = x_{best} + \varepsilon A^t \oplus L(\lambda) \tag{24}$$

Based on these idealizations, the basic steps of BA can be described as a general pseudo-code shown in Fig. 5. In Fig. 5, if the concept of Lévy flights in lines 11, 12 and mutation (lines 17–22) are omitted, then Fig. 5 corresponds to the general pseudo-code for simple BA. If only the concept of mutation (lines 17–22) is not used in Fig. 5, then it corresponds to the pseudo-code for LBA, otherwise Fig. 5 shows the general pseudo-code for LBAM algorithm.

3.4 Cuckoo search algorithm and its modified form

Yang and Deb (2010b), Gandomi et al. (2013), Yang and Deb (2014), inspired by brood parasitism of some cuckoo species, developed a nature-inspired metaheuristic optimization algorithm called Cuckoo search algorithm (CSA). In addition, CSA algorithm is enhanced by the Lévy flights trajectory of some birds, rather than by simple random walks. For describing this algorithm, Yang et al. uses the following three idealized rules:

1. Each Cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
2. The best nest with high quality of eggs (solution) are carried over to the next iterations;

Fig. 5 General pseudo-code for MBA

```
Modified Bat Algorithm (MBA)
Begin
  /* MBA parameter initialization */
    Define objective function f (X);
    Generate initial bat population x_i and initial velocities v_i (i = 1,
    2,...n);
    Define pulse frequency f_i at x_i;
    Initialize pulse rates r_i and the loudness A_i;
  /* End of MBA parameter initialization */
  While not t                              /* t is a termination criterion */
     Generate new solutions by adjusting frequency, and updating
     velocities and locations/solutions by Lévy flights;
     If (rand > r_i)
        Select a solution among the best solutions;
        Generate a local solution around the selected best solution;
     Else
       /* Mutation */
           Compute mutation rate probability MR_i via equation (7);
           If (MR_i < rand)
              Perform mutation via equation (8);
           End if
       /* End of Mutation */
     End if
     Generate a new solution by flying randomly;
     If (rand < r_i and f (x_i) < f (x*))
         Accept the new solutions;
         Increase r_i and reduce A_i;
     End if
     Rank the bats and find the current best;
   End while
   Postprocess results and visualization;
End
```

3. The number of available host nests is fixed, and a host can discover an alien egg with probability $p_a \in [0,1]$. In this case, the host bird can either throw the egg away or simply abandon the nest so as to build a completely new nest in a new location.

For simplicity, the last assumption can be approximated by a fraction p_a of the n host nests being replaced by new nests (with new random solutions). For a maximization problem, the quality i.e. fitness of a solution can simply be proportional to the value of the objective function. When new solutions x^t are generating for, say, a cuckoo i, a Lévy flight is performed as approximated by (Iglesias et al. 2018):

$$x_i^t = x_i^{t-1} + \alpha \oplus L(\lambda) \tag{25}$$

where $\alpha > 0$ is the step size, which should be related to the scale of the specified problem.

As authors in Yang and Deb (2010b), already introduced the Lévy flights distribution concept to enhance the performance, so only mutation strategy is applied to simple CSA to explore the search space. The new modified algorithm so formulated is named as *Cuckoo search algorithm with mutation* (CSAM). The basic steps of CSAM can be summarized as the pseudo-code shown in Fig. 6. If the concept of mutation (lines 9 to 14) is withdrawn from Fig. 6, then it corresponds to general pseudo-code for simple CSA.

Fig. 6 General pseudo-code for CSAM

```
Cuckoo Search Algorithm with Mutation (CSAM)
Begin
  /* CSAM parameter initialization */
     Define objective function f(X);
     Generate initial population of n host nests x_i (i = 1, 2,...,n);
  /* End of CSAM parameter initialization */
  While not t                              /* t is a termination criterion */
     Get a cuckoo (say i) randomly by Lévy flights;
     /* Mutation */
        Compute mutation rate probability MR_i via equation (7);
        If (MR_i < rand)
           Perform mutation via equation (8);
        End if
     /* End of mutation */
     Evaluate its quality/fitness F_i;
     Choose a nest among n (say j) randomly;
     If (F_i > F_j),
        Replace j by the new solution;
     End if
     A fraction (p_a) of worse nests are abandoned and new ones
     are built;
     Keep the best solutions (or nests with quality solutions);
     Rank the solutions and find the current best;
  End while
  Postprocess results and visualization;
End
```

3.5 Flower pollination algorithm and its modified form

Yang (2012b) and Yang et al. (2014), inspired by the flow pollination process of flowering plants, introduced an optimization algorithm called Flower pollination algorithm (FPA). For describing this metaheuristic algorithm, the following four idealized rules were used:

1. For global pollination process, biotic cross-pollination is used and pollen-carrying pollinators obey Lévy flights distributions.
2. For local pollination, abiotic and self-pollination are used.
3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
4. The interaction of local pollination and global pollination can be controlled by a switch probability $p \in [0,1]$, with a slight bias towards local pollination.

In FPA, the global pollination and local pollination are two main steps. In the global pollination step, flower pollens are carried by pollinators such as insects, and pollens can travel over a long distance because insects can often fly and travel over a much longer range. The first rule and flower constancy (i.e. third rule) can be written mathematically into a single equation (Yang 2012b):

$$x_i^t = x_i^{t-1} + \gamma L(\lambda)(x_* - x_i^{t-1}) \tag{26}$$

where x_i^t is the pollen i or solution vector x_i at iteration t, x_* is the current best solution (i.e. most fittest) found among all solutions at the current iteration and γ is a scaling factor to control the step size. The Lévy flight based step size $L(\lambda)$ corresponds to strength of the pollination. Since insects may travel over a long distance with various distance steps, a

Lévy flight can be used to mimic this characteristic efficiently. That is, $L > 0$ is drawn from a Lévy flight distribution.

For local pollination, the second rule and flower constancy can be written mathematically by a single equation:

$$x_i^t = x_i^{t-1} + \in (x_j^{t-1} - x_k^{t-1}) \tag{27}$$

where x_j^{t-1} and x_k^{t-1} are pollens from different flowers of the same plant species. This essentially mimics the flower constancy in a limited neighborhood. Mathematically, if x_j^t and x_k^t are selected from the same population, this become a local random walk if $\in$ is drawn from a uniform distribution in [0,1]. Pollination may also occur in a flower from the neighboring flower than by the far away flowers. For this, a switch probability (i.e. fourth rule) or proximity probability p can be used to switch between global pollination and local pollination.

Like CSA, the author in Yang (2012b) already introduced the concept of Lévy flight distributions in FPA, so only mutation based on fitness value is added to simple FPA. The new algorithm so formed is named in this paper as *Flower pollination algorithm with mutation* (FPAM) which is summarized as pseudo-code shown in Fig. 7. The only difference in the pseudo-code for FPA and FPAM is only the addition of mutation (lines 19–24) in Fig. 7. If lines 19–24 are not used in Fig. 7 then it corresponds to the general pseudo-code for simple FPA.

Fig. 7 General pseudo-code for FPAM

1. Flower Pollination Algorithm with Mutation (FPAM)
2. Begin
3. **/* FPAM parameter initialization */**
4. Define objective function $f(X)$;
5. Initialize a population of n flowers/pollen gametes with random
6. solutions;
7. Find the best solution g_* in the initial population;
8. Define a switch probability $p \in [0,1]$;
9. **/* End of FPAM parameter initialization */**
10. **While** not t /* t is a termination criterion */
11. **For** $i = 1 : n$ /*all n flowers */
12. **If** ($rand < p$)
13. Draw a (d–dimensional) step vector L via Lévy flights;
14. Perform global pollination via equation (26);
15. **Else**
16. Draw $\in$ from a uniform distribution in [0,1];
17. Perform local pollination via equation (27);
18. **End if**
19. **/* Mutation */**
20. Compute mutation rate probability MR_i via equation (7);
21. **If** ($MR_i < rand$)
22. Perform mutation via equation (8);
23. **End if**
24. **/* End of mutation */**
25. Evaluate new solutions;
26. If new solutions are better, update them in the population;
27. **End for**
28. Rank the solutions and Find the current best solution x_*;
29. **End while**
30. Postprocess results and visualization;
31. End

4 Finding near-OGRs: problem formulation

Both simplicity and efficiency attracts researchers towards natural phenomenon to solve NP–complete and complex optimization problems. The first problem investigated in this research is to find Golomb rulers for unequal channel allocation. Second problem is to obtain either optimal or near-OGRs through nature-inspired metaheuristic algorithms by optimizing the ruler length so as to conserve the total occupied optical channel bandwidth.

If each individual element in an obtained set (i.e. non-negative integer location) is a Golomb ruler, the sum of all elements of an individual set forms the total occupied optical channel bandwidth. Thus if the spacing between any pair of channels in a Golomb ruler set G is denoted as CS, an individual element is as IE and the total number of channels/marks is n, then the ruler length RL and the total optical channel bandwidth TBW are approximated by the following equations:

Ruler Length (*RL*):

$$RL = \sum_{i=1}^{n-1} (CS)_i \tag{28a}$$

subject to $(CS)_i \neq (CS)_j$

Alternatively, Eq. (28a) can also be approximated as:

$$RL = IE(n) - IE(1) \tag{28b}$$

Total Bandwidth (*TBW*):

$$TBW = \sum_{i=1}^{n} (IE)_i. \tag{29}$$

subject to $(IE)_i \neq (IE)_j$

where $i, j = 1, 2, \dots, n$ with $i \neq j$ are distinct in both Eqs. (28) and (29).

4.1 Nature-inspired algorithms to find near-OGRs

The general pseudo-code to find near-OGRs by using nature-inspired optimization algorithms proposed in this paper is shown in Fig. 8. The core of the proposed algorithms is lines 19–30 which find Golomb rulers for a number of iterations or until either an optimal or near–to–optimal solution is found. Also the size of the generated population must be equal at the end of iteration to the initial population size (*Popsize*). Since there are many solutions, a replacement strategy must be performed as shown in Fig. 8 to remove the worst individuals. This mean that the proposed algorithms maintain a fixed population of rulers and performs a fixed number of iterations until either an optimal or near–to–optimal solution is found.

```
Nature–Inspired Optimization Algorithms to Find Near–OGRs
Begin
    /* Parameter initialization */
        Define operating parameters for nature–inspired optimization algorithms;
        Initialize the number of channels, lower and upper bound on the ruler length;
        While not Popsize                          /* Popsize is the population size input by the user */
            Generate a random set of candidates (integer population);
                            /* Number of integers in candidates is being equal to the number of channels */
            Check Golombness of each candidates;
            If Golombness is satisfied
                Retain that candidate;
            Else
                Remove that particular candidate from the generated population;
            End if
        End while
        Compute the fitness values;      /* fitness value represents the cost value i.e. ruler length and total optical channel bandwidth */
        Rank the candidates from best to worst based on fitness values;
    /* End of parameter initialization */
    While not t                                    /* t is a termination criterion */
  A:    Call any nature–inspired optimization algorithm to determine new optimal set of candidates;
        Recheck Golombness of updated candidates;
        If Golombness is satisfied
            Retain that candidate and then go to B;
        Else
            Retain the previous generated candidate and then go to A;
                  /* Previous generated candidate is being equal to the candidate generated into the parameter initialization step*/
        End if
  B:    Recompute the fitness values of the modified candidates;
        Rank the candidates from best to worst based on fitness values and find the current best;
    End while
    Display the near–OGR sequences;
End
```

Fig. 8 General pseudo-code to find near-OGR sequences by using nature-inspired optimization algorithms

5 Simulation results

This section presents the performance of proposed optimization algorithms and their performance comparison with best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html), two of the conventional computing algorithms i.e. EQC and SA (Kwong and Yang 1997; Randhawa et al. 2009) and three nature-inspired algorithms i.e. GAs (Bansal 2014), BBO (Bansal 2014) and simple BB–BC (Bansal and Sharma 2017), of finding unequal channel spacing. All the proposed algorithms to find near-OGRs have coded and tested in Matlab–R2017a language running under Windows 7, 64–bit operating system. The algorithms have been executed on Intel(R) core™ 2 Duo CPU T6600 @ 2.20 GHz processor Laptop with a RAM of 3 Gb and hard drive of 320 Gb.

5.1 Parameters selection for the proposed algorithms

To find either optimal or near-optimal solutions after a number of careful experimentation, following optimum parameter values of proposed algorithms have finally been settled as shown

Table 1 Simulation parameters for MBB–BC

Parameter	Value
c_1	0.1
c_2	5
p_m	0.05

Table 2 Simulation parameters for FA and MFA

Parameter	Value
α	0.5
β	0.2
γ	1.0
p_m	0.05

Table 3 Simulation parameters for BA and MBA

Parameter	Value
A^0	0.8
r^0	0.5
p_m	0.01

Table 4 Simulation parameters for CSA and CSAM

Parameter	Value
α	0.01
p_a	0.5
p_m	0.05

Table 5 Simulation parameters for FPA and FPAM

Parameter	Value
γ	1.0
p	0.8
p_m	0.01

in Tables 1, 2, 3, 4 and 5. The selection of a suitable parameter values for nature-inspired algorithms are problem specific as there are no concrete rules.

5.2 Near-OGR sequences

With the above mentioned parameters setting, the large numbers of sets of trials for various marks/channels were conducted. Each algorithm was executed 20 times until near-optimal solution was found. The generated near-OGRs for different marks by proposed metaheuristic algorithms are shown in "Appendix". It has been verified that all the generated sequences are Golomb rulers. Although the proposed metaheuristic algorithms find same near-OGRs, but the difference is in required maximum number of iterations and computational time which is discussed in the following subsections.

5.3 Influence of selecting different population size on the performance of proposed algorithms

In this subsection, the influence of selecting different *Popsize* on the performance of proposed optimization algorithms for different values of channels is examined. The increased *Popsize* increases the diversity of potential solutions, and helps to explore the search space. But as the *Popsize* increase, the computation time required to get either the optimal or near-optimal solutions increase slightly as the diversity of possible solutions increase. But after some limit, it is not useful to increase *Popsize*, because it does not help in solving the problem faster. The choice of the best *Popsize* for nature-inspired optimization algorithms depends on the type of the problem (Bansal 2019).

Golomb rulers realized from 10- to 16-marks by TS (Cotta et al. 2006), maximum *Popsize* set was 190. The hybrid approach proposed in (Ayari et al. 2010) to find Golomb rulers from 11- to 23-marks, the *Popsize* was set between 20 and 2000. The near-OGRs found by the HE algorithms (Dotú and Hentenryck 2005), maximum *Popsize* set was 50. For the algorithms GAs and BBO (Bansal 2014), to find near-OGRs, maximum *Popsize* set was 30.

With the above mentioned parameter settings, the experiment with *Popsize* varying from 10 to 100 for all the proposed metaheuristic algorithms was performed. It was noted that *Popsize* has little significant effect on the performance of all proposed metaheuristic algorithms. By carefully looking at the results, the *Popsize* of 10 in all proposed algorithms was found to be adequate for finding near-OGRs.

5.4 Influence of increasing iterations on total optical channel bandwidth

The choice of the best maximum iteration for metaheuristic algorithms is always critical for specific problems. Increasing the numbers of iteration, will increase the possibility of reaching optimal solutions and promoting the exploitation of the search space so as to obtain the global optimization abilities of the metaheuristic algorithms. This will maintain a better population diversity and the global search ability is improved, which efficiently improves the accuracy and efficiency of the metaheuristic algorithms. This is because after a number of iteration, it is necessary to search from a different path so as to get improved solutions. The algorithm effectively jumps out of the local optimal value and continues to optimize. This means, the chance to find the correct search direction increases considerably.

In this subsection the influence of increasing the number of iterations on proposed algorithms with the same parameter settings as mentioned above subsections is examined. By increasing the number of iterations, the total optical channel bandwidth tends to decrease; it means that the rulers reach their optimal or near-optimal values after certain iterations. This is the point where no further improvement is seen. Figure 9 illustrates the influence of increasing iterations on the performance of proposed algorithms for various channels. It is noted that the iterations have little effect for low value marks. But for higher order marks, the iterations have a great effect on the total optical channel bandwidth i.e. total optical bandwidth gets optimized after a certain numbers of iterations.

In literatures (Cotta et al. 2006) and (Ayari et al. 2010), the maximum numbers of iterations (*Maxiter*) for TS algorithm to find Golomb rulers were set to 10000 and 30000 respectively. The hybrid approach proposed in (Ayari et al. 2010) to find Golomb rulers

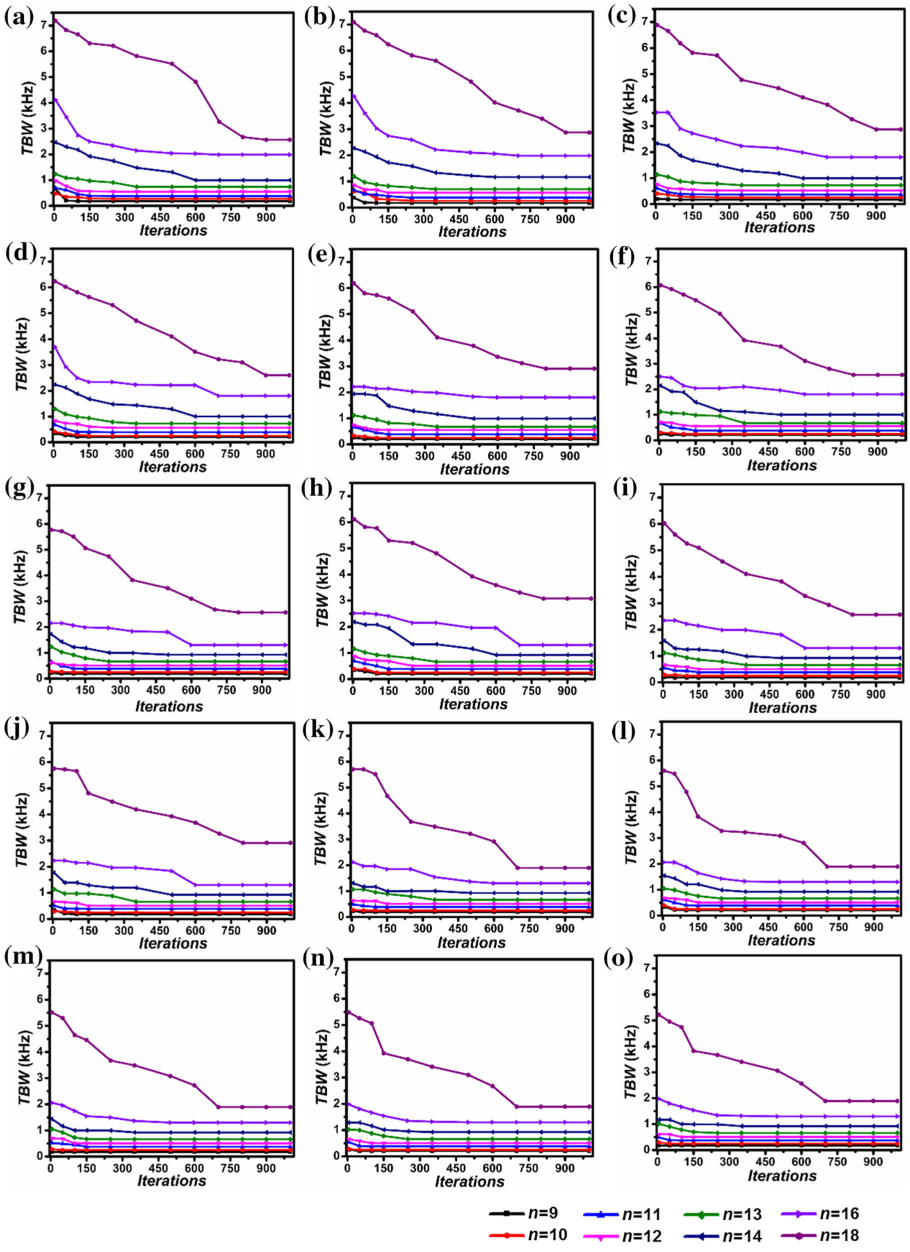


Fig. 9 Influence of iterations on *TBW* for **a** BB-BCM; **b** LBB-BC; **c** LBB-BCM; **d** FA; **e** FAM; **f** LFA; **g** LFAM; **h** BA; **i** BAM; **j** LBA; **k** LBAM; **l** CSA; **m** CSAM; **n** FPA; and **o** FPAM for various channels

the maximum number of iterations set were 100000. In (Bansal 2014), it was noted that to find near-OGRs, hybrid evolutionary algorithms (Dotú and Hentenryck 2005) get stabilized in and around 10000 iterations, while GAs and BBO algorithms stabilized in and around 5000 iterations. By carefully looking at the results, it is concluded that all the proposed optimization algorithms in this paper to find either optimal or near-OGRs stabilized in or around 1000 iterations. To find n-channel near-OGRs, the value of *Maxiter* parameter during the simulations was selected to 50 times the number of channel (n), i.e. $Maxiter = 50 \times n$. In summary, the FPAM algorithm shows stronger optimization performance and higher optimization efficiency and faster convergence rate than the other algorithms.

5.5 Performance comparison of proposed algorithms with previous existing algorithms in terms of ruler length and total optical channel bandwidth

Table 6 enlists the ruler length and total occupied channel bandwidth by different sequences obtained from the proposed algorithms after 20 executions and their performance comparison with best known OGRs (best solutions), EQC, SA, GAs, BBO and simple BB–BC. According to (Kwong and Yang 1997), the applications of EQC and SA is 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 6. Comparing the experimental results obtained from the proposed algorithms with best known OGRs and existing algorithms, it is noted that there is a significant improvement in the ruler length and thus the total occupied channel bandwidth that is, the results gets better.

From Table 6, it is also observed that simulation results are particularly impressive. First observe that for all the proposed algorithms, the ruler length obtained up to 13-marks is same as that of best known OGRs and the total optical channel bandwidth occupied for marks 5 to 9 and 11 is smaller than the best known OGRs, while all the other rulers obtained are either optimal or near-optimal. Second observe that the algorithms BB–BCM and LBB–BC do not find best known rulers after 7-marks, but finds near-optimal rulers for 8- to 20-marks. Algorithm LBB–BCM can find best optimal rulers up to 8-marks, but finds near-optimal rulers after 8-marks. FA can find best rulers for up to 11-marks. Algorithms FAM and LFA can find best rulers for up to 12-marks and near-optimal rulers after 12-marks. By combining algorithms FAM and LFA into a single algorithm named LFAM, best OGRs up to 16-marks and near-optimal rulers for 17- to 20-marks can be find efficiently. BA, BAM and LBA can find best rulers up to 17-marks and near-optimal rulers for 18- to 20-marks. The algorithms LBAM, CSA, CSAM, FPA and FPAM can find best rulers up to 20-marks very efficiently and effectively in a reasonable computational time.

From simulation results, it is concluded that modified forms of the proposed algorithms to find near-OGRs, slightly outperforms the algorithms presented in their simplified forms. As illustrated in Table 6 for higher order marks, the algorithms CSA, FPA, BA and their modified forms outperforms the other algorithms in terms of both the ruler length and total occupied channel bandwidth. Figure 10 shows the graphical representation of Table 6.

5.6 Performance comparison of proposed algorithms in terms of computational time

Finding Golomb rulers is an extremely challenging optimization problem. The OGRs generation by exhaustive parallel search algorithms for higher order marks is computationally very time consuming, which took several hours, months, even years of calculation on the network of several thousand computers (Shearer 1998; Rankin 1993; Dollas et al. 1998; http://www.distributed.net/ogr). For example, rulers with 20- to 26-marks were found by

Table 6 Performance comparison of proposed nature-inspired metaheuristic algorithms to channel allocation

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms									
			Conventional algorithms				Existing nature-inspired algorithms					
			EQC (Kwong and Yang 1997; Randhawa et al. 2009)		SA (Kwong and Yang 1997; Randhawa et al. 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal and Sharma 2017)	
	RL	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)
3	3	4	6	10	6	4	3	4	3	4	3	4
4	6	11	15	28	15	28	6 7	11	6 7	11	6 7	11
5	11	25 28	–	–	–	–	12 13	23 25 29	12 13	23 24	11 12	23 25
6	17	44 47 50 52	45	140	20	60	17 18 21	42 44 45	17 18 20 21	42 43 44 45 49	17 18	42 44
7	25	77 81 87 90 95	–	–	–	–	27 28 29 30 31 32	73 78 79 80 83 86 95	27 29 30 31 32 33	73 82 83 84 91 95	25 26 28 30	73 74 77 81

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms									
			Conventional algorithms				Existing nature-inspired algorithms					
			EQC (Kwong and Yang 1997; Randhawa et al. 2009)		SA (Kwong and Yang 1997; Randhawa et al. 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal and Sharma 2017)	
	RL	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)
8	34	117	91	378	49	189	35 41 42 45 46	121 126 128 129 131 133	34 39 40 42	121 125 127 131	39 41 42	113 118 119
9	44	206	–	–	–	–	52 56 59 61 63 65	192 193 196 203 225	49 56 61 62 64	187 200 201 206 215	44 45 46 61	179 248 253 262
10	55	249	–	–	–	–	75 76	283 287 301	74	274	77	258
11	72	386 391	–	–	–	–	94 96	395 456	86 103 104 114 118	378 435 440 491	72 105	377 490 456

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms									
			Conventional algorithms				Existing nature-inspired algorithms					
			EQC (Kwong and Yang 1997; Randhawa et al. 2009)		SA (Kwong and Yang 1997; Randhawa et al. 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal and Sharma 2017)	
	RL	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)
12	85	503	231	1441	132	682	123 128 137	532 581 660	116 124 138	556 590 605	85 91	550 580 613
13	106	660	–	–	–	–	203 241	1015 1048	156 171 187	768 786 970	110 113	768 753
14	127	924	325	2340	286	1820	206 228 230	1172 1177 1285	169 206 221	991 1001 1166	221	1166
15	151	1047	–	–	–	–	275 298	1634 1653	260 267	1322 1554	267	1322
16	177	1298	–	–	–	–	316	1985	283	1804	316	1985
17	199	1661	–	–	–	–	355	2205	354 369	2201 2208	369	2201
18	216	1894	561	5203	493	5100	427 463	2599 3079	362 445	2566 2912	427	3079
19	246	2225	–	–	–	–	567 597	3432 5067	467 475 584	3337 3408 4101	584	4101

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms									
			Conventional algorithms				Existing nature-inspired algorithms					
			EQC (Kwong and Yang 1997; Randhawa et al. 2009)		SA (Kwong and Yang 1997; Randhawa et al. 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal and Sharma 2017)	
	RL	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)
20	283	2794	703	7163	703	6460	615 673 680 691	4660 4826 4905 4941	578 593 649	4306 4517 4859	691	4941

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms															
			Proposed nature-inspired algorithms															
			BB–BCM		LBB–BC		LBB–BCM		FA		FAMFAM		LFA		LFAM		BA	
	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	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms																
			Proposed nature-inspired algorithms																
			BB–BCM		LBB–BC		LBB–BCM		FA		FAMFAM		LFA		LFAM		BA	
	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)
4	6	11	6	11	6	11	6	11	6	11	6	11	6	11	6	11	6	11
			7		7		7		7		7		7		7		7	
5	11	25	11	23	11	23	11	23	11	23	11	23	11	23	11	23	11	23
		28	12	25	12	24	12	25	12	24	12	24	12	24	12	24	12	25
					13	25		28	13	25	13				13		13	
6	17	44	17	42	17	42	17	42	17	42	17	42	17	42	17	42	17	42
		47	18	44	18	44	18	44	18	44	18	44	18	44	18	44	18	44
		50		46				46										
		52																
7	25	77	25	73	25	73	25	73	25	73	25	73	25	73	25	73	25	74
		81	27	74	30	77	28	74	26	77	27	74	26	77	26	74	28	77
		87	28	77		81	30	77	27	80	28	77	27	80	27	77		81
		90	30	79						81				81	28			90
		95																
8	34	117	39	113	39	113	34	113	34	113	34	113	34	113	34	113	34	113
			41	118	41	118	39	117	39	117	39	117	39	117	39	117	39	117

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms															
			Proposed nature-inspired algorithms															
			BB–BCM		LBB–BC		LBB–BCM		FA		FAMFAM		LFA		LFAM		BA	
	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)
9	44	206	44 57	183 215 226	46 47 58	177 204 217 228	44 55	176 208	44 49	206 208	44 49	206	44 49	206	44 47 49	185 206	44	206
10	55	249	55 58 74	274 299 311	55 77	258 321 341	55	259 309	55	249	55	249	55	249	55	249	55	249
11	72	386 391	72 105	377 435	72 103	378 439	72 96	369 397 434	72	391	72	386	72 103	378 391	72 103	378 386 391	72	386
12	85	503	85 90	549 565	85 90	565 567	85 91 93	520 551 550	85	515	85	503	85	503	85	503	85	503
13	106	660	106 110	736 755 848	109 111	700 751 763	106 111	725 744	106	725 744	106 111	675 725	106 111	673 720	106	660	106	660

Table 6 (continued)

n	Best known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms: Proposed nature-inspired algorithms															
			BB–BCM		LBB–BC		LBB–BCM		FA		FAMFAM		LFA		LFAM		BA	
	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)
14	127	924	229	996	221	1166	206 226	993 1285	169 206	991 1001	206	991	169	1001	127	924	127	924
15	151	1047	267	1322	267	1322	260	1554	260	1554	151	1047	151	1047	151	1047	151	1047
16	177	1298	316	1985	316	1985	283	1804	283	1804	283	1804	283	1804	177	1298	177	1298
17	199	1661	369	2201	369	2201	355	2205	355	2205	354	2208	354	2208	369	2201	199	1661
18	216	1894	445	2566	436	2872	436	2872	463	2599	362	2912	445	2566	445	2566	427	3079
19	246	2225	567	3432	584	4101	467	3337	567	3432	467	3337	475	3408	467	3337	467	3337
20	283	2794	673	4826	673	4826	649	4517	649	4517	615	4660	615	4660	578	4306	578 615	4306 4660

Table 6 (continued)

n	Best Known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms													
			Proposed nature-inspired algorithms													
			BAM		LBA		LBAMLBAM		CSA		CSAM		FPA		FPAM	
	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	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	6	11	6 7	11	6 7	11	6	11	6 7	11	6	11	6 7	11	6	11
5	11	25 28	11 12 13	23 24	11 12 13	23 24 28	11 12 13	23 24 25	11 12	23 24	11 12	23 24	11 12	23 24	11 12	23 24
6	17	44 47 50 52	17 18	42 44	17 18	42 44 47 50	17 18	42 44 47 50	17 18	42 44 47 50	17 18	42 44 47	17 18	42 44 47	17 18	42 44 47 50
7	25	77 81 87 90 95	25 26 27	73 77	25 27	73 77	25 26 27	73 77	25 26 27	73 77 80	25 26 27	73 77	25 26 27	73 77 80 81	25 26 27	73 77
8	34	117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117	34 39	113 117

Table 6 (continued)

n	Best Known OGRs (Bloom and Golomb 1977; Shearer 1990; Rankin 1993; Colannino 2003; Dollas et al. 1998; http://mathworld.wolfram.com/GolombRuler.html)		Algorithms: Proposed nature-inspired algorithms: BAM		LBA		LBAMLBAM		CSA		CSAM		FPA		FPAM	
	RL	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)	*RL*	*TBW* (Hz)
9	44	206	44 57	183 206	44 58	177 206	44 47	185 206	44	206	44 47 55	176 185 206	44	206	44 47 55	176 185 206
10	55	249	55	249	55	249	55	249	55	249	55	249	55	249	55	249
11	72	386 391	72	386	72	391	72	386 391	72	391	72 103	378 386	72	386	72 103	378 386
12	85	503	85	503	85	503	85	503	85	503	85	503	85	503	85	503
13	106	660	106	660	106	660	106	660	106	660	106	660	106	660	106	660
14	127	924	127	924	127	924	127	924	127	924	127	924	127	924	127	924
15	151	1047	151	1047	151	1047	151	1047	151	1047	151	1047	151	1047	151	1047
16	177	1298	177	1298	177	1298	177	1298	177	1298	177	1298	177	1298	177	1298
17	199	1661	199	1661	199	1661	199	1661	199	1661	199	1661	199	1661	199	1661
18	216	1894	445	2566	362	2912	216	1894	216	1894	216	1894	216	1894	216	1894
19	246	2225	467	3337	475	3408	246	2225	246	2225	246	2225	246	2225	246	2225
20	283	2794	578	4306	578 593	4306 4859	283	2794	283	2794	283	2794	283	2794	283	2794

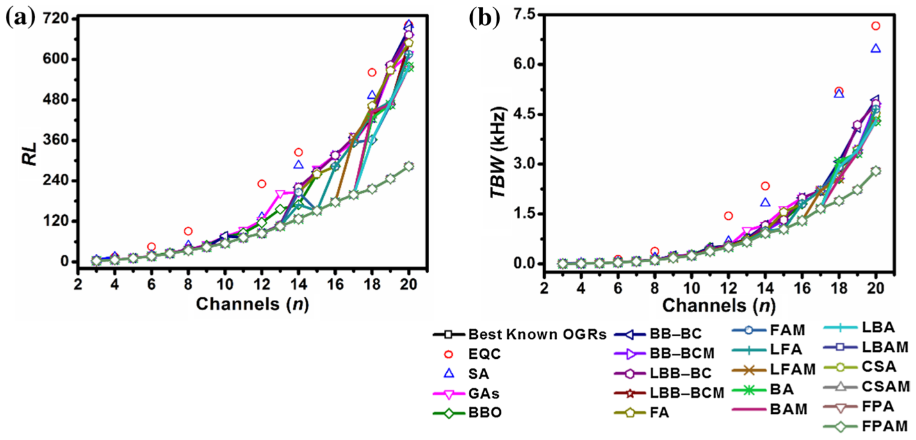
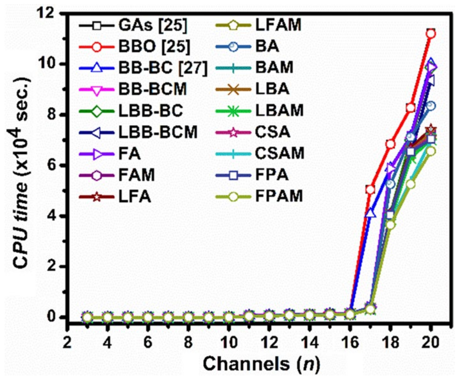


Fig. 10 Comparison of proposed algorithms in terms of **a** *RL* and **b** *TBW* with the other existing algorithms

Fig. 11 Comparison of average *CPU time* taken by proposed algorithms for various channels



distributed OGR project (http://www.distributed.net/ogr) which took several years of calculations on many computers to prove the optimality of the rulers.

This subsection is devoted to report the experimental average *CPU time* taken to find either optimal or near-OGRs by the proposed algorithms and their comparison with the computation time taken by existing algorithms (Shearer 1990; Rankin 1993; Soliday et al. 1995; Ayari et al. 2010; Bansal 2014, 2017; Bansal and Sharma 2017; Dollas et al. 1998; http://www.distributed.net/ogr). Figure 11 illustrates the average *CPU time* taken by proposed metaheuristic algorithms to find near-OGRs up to 20-marks. In (Soliday et al. 1995), it is identified that to find Golomb rulers from heuristic based exhaustive search algorithm, the times varied from 0.035 s to 6 weeks for 5- to 13-marks ruler, whereas by non-heuristic exhaustive search algorithms took approximately 12.57 min for 10-marks, 2.28 years for 12-marks, 2.07×10^4 years for 14-marks, 3.92×10^9 years for 16-marks, 1.61×10^{15} years for 18-marks and 9.36×10^{20} years for 20-marks ruler. In (Ayari et al. 2010), it is reported that *CPU time* taken by TS algorithm to find OGRs is around 0.1 s for 5-marks, 720 s for 10-marks, 960 s for 11-marks, 1913s for 12-marks and 2516 s (around 41 min) for 13-marks. The OGRs realized by HGA (Ayari et al. 2010) took around 5 h for 11-marks, 8 h for 12-marks, and 11 h for 13-marks. The OGRs realized by the exhaustive search algorithms in (Shearer 1990) for 14- and 16-marks, took nearly one hour and hundred hours respectively, while 17-, 18- and 19-marks OGRs realized in (Rankin 1993) and (Dollas et al. 1998), 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 algorithms GAs and BBO (Bansal 2014), the maximum execution time was approximately 31 h i.e. nearly 1.3 days, while for BB–BC (Bansal and Sharma 2017) the maximum execution time was around 28 h i.e. almost 1.1 days.

It is noted that for proposed algorithms, the average *CPU time* varied from 0.000 s for 3-marks ruler to approximately 27 h for 20-marks ruler. The maximum and minimum execution time taken by the proposed algorithms for 20-marks ruler is about 27 and 19 h, respectively. By introducing the concept of mutation and

Lévy flight strategies with the proposed algorithms, the minimum execution time is reduced to approximately 18 h i.e. less than one day. This represents the improvement achieved by the use of proposed optimization algorithms and their modified forms to find near-OGRs. From Fig. 11, it is further observed that algorithm FPAM outperforms the other algorithms in terms of computational time.

5.7 Maximum computation complexity of proposed algorithms in terms of big O notation

The nature-inspired metaheuristic algorithms are stochastic process that execute randomly operations. For this reason, it is not practical to conduct a complexity analysis from a deterministic point of view. However, it is possible to have an idea of this complexity through the mathematical notation called Big O notation.

To find the optimal solutions, the proposed metaheuristic algorithms have an initialization stage and a subsequent stage of iterations. The computational complexity of nature-inspired algorithm depends upon n, *Popsize* and *Maxiter*:

$$\text{Computation complexity} = \mathrm{O}(n \times Popsize \times Maxiter) \tag{30}$$

For all the proposed metaheuristic algorithms, the maximum computational complexity in terms of Big O notation is

$$\text{Computation complexity} = \mathrm{O}(n \times 10 \times 50 \times n) = \mathrm{O}\left(500n^2\right) \tag{31}$$

Thus the computational complexity for all the proposed metaheuristic algorithms is directly proportional to the square of the input mark/channel value.

Table 7 Comparison of Wilcoxon rank-sum analysis for FPAM and other algorithms

Algorithm	*RL*		*TBW* (Hz)		*CPU time* (Sec.)	
	p value	*h*	*p* value	*h*	*p* value	*h*
FPAM versus GAs	0.0001	1	< 0.0001	1	< 0.0001	1
FPAM versus BBO	0.0002	1	< 0.0001	1	< 0.0001	1
FPAM versus BB-BC	0.002	1	< 0.0001	1	< 0.0001	1
FPAM versus BB-BCM	0.0078	1	< 0.0001	1	0.0001	1
FPAM versus LBB-BC	0.0156	1	< 0.0001	1	0.0001	1
FPAM versus LBB-BCM	0.00156	1	0.0001	1	0.0001	1
FPAM versus FA	0.0156	1	0.0001	1	< 0.0001	1
FPAM versus FAM	0.0313	1	0.0005	1	0.0001	1
FPAM versus LFA	0.0313	1	0.0039	1	0.0001	1
FPAM versus LFAM	0.125	0	0.0625	0	0.0001	1
FPAM versus BA	–	–	0.0625	0	< 0.0001	1
FPAM versus BAM	–	–	–	–	0.0001	1
FPAM versus LBA	–	–	–	–	< 0.0001	1
FPAM versus LBAM	–	–	–	–	0.0001	1
FPAM versus CSA	–	–	–	–	0.0001	1
FPAM versus CSAM	–	–	–	–	0.0001	1
FPAM versus FPA	–	–	–	–	0.0001	1

5.8 Wilcoxon rank-sum test of proposed algorithms

In order to validate the proposed algorithms, the non-parametric statistical analysis with 5% significance level is conducted to analyze and rank the algorithms. With the non-parametric statistical analysis, we can make sure that the results are not caused by chance (Li et al. 2019). The Wilcoxon rank-sum statistical analysis (García et al. 2009) is performed on *RL*, *TBW* and *CPU time*. The Wilcoxon statistical comparison analysis between FPAM and other algorithms are listed in Table 7. Since the algorithms BA, CSA, FPA and their improved forms find same near-OGR sequences at different computational time so the *RL* and *TBW* p value is shown by a dash line in Table 7. In this experiment p value < 0.05 and $h = 1$ indicate that the difference between the two data is significant.

6 Conclusions and future scope

In this paper, WDM channel allocation algorithm by considering the concept of OGRs is presented. Finding OGRs through conventional computing algorithms is computationally hard problem because as the number of marks increases, the search for OGRs rises exponentially. The aim to use metaheuristic algorithms is not necessarily to produce perfect solutions, but to produce the near–to–optimal solutions under the given constraints. Even if exact algorithms are able to find optimal/near-OGRs, they remain unpractical in terms of computational time. This paper presented the application of five nature-inspired metaheuristic algorithms (BB–BC, FA, BA, CSA and FPA) to solve near-OGRs problem. The main technical contribution of this paper was to modify the nature-inspired metaheuristic algorithms by applying mutation and Lévy flight strategies. The proposed algorithms have been validated and compared with other existing algorithms to find near-OGRs. It has been observed that modified forms finds near-OGRs very efficiently and effectively than their simplified forms. The enumerated near-OGRs were compared with those enumerated through existing conventional and nature-inspired algorithms in terms of iterations, ruler length, total optical channel bandwidth and computational time. Simulations and comparison show that the proposed algorithms are superior to the existing algorithms. From preliminary results it is also concluded that for large order marks, MFA outperforms FA and MBB–BC, MBA outperforms MFA and BA, CSAM outperforms MBA and CSA, while FPAM is slightly outperforms CSAM and FPA in terms of ruler length, total channel bandwidth, experimental computation time and maximum number of iterations needed to find near-OGRs. The results obtained from simulation experiment and Wilcoxon rank-sum analysis show that the proposed FPAM is potentially more superior in terms of exploitation and exploration abilities, success rate and convergence speed compared with the other algorithms in solving such a NP–complete problem.

To date, none of the researchers show the implementation of their algorithm in real optical WDM systems in order to see the complexity of realizing the unequal channel spacing. Although numerous algorithms have been suggested for finding near-OGRs, yet there is no uniformly accepted formulation. So, in order for these algorithms to be of practical use, it is desired that the performance of these algorithms for higher order OGRs up to about several thousand channels may be evaluated and may be used to provide unequal channel spacing in real WDM system. Though this process will be very time consuming yet this needs be done for this work to be of some use in the field of communication engineering.

Appendix

Tables 8 and 9 lists the near-OGRs found by proposed algorithms for various marks.

Table 8 Near-OGRs found by MBB–BC, FA and MFA

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3
4	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6
	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7
5	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11
	12	0 1 3 7 12	12	0 1 3 7 12	11	0 2 7 8 11	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12
			13	0 1 4 6 13	12	0 1 3 7 12	12	0 1 3 8 12	13	0 1 4 6 13			13	0 1 4 6 13
							13	0 1 4 6 13						
6	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17
	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18
	18	0 1 5 7 15 18			18	0 1 5 7 15 18								
7	25	0 1 4 10 18 23 25	25	0 2 3 10 16 21 25	25	0 2 3 10 16 21 25	25	0 2 6 9 14 24 25	25	0 2 3 10 16 21 25	25	0 2 6 9 14 24 25	25	0 2 3 10 16 21 25
	27	0 1 5 7 15 24 27	25	0 1 4 10 18 23 25	28	0 1 3 8 12 22 28	25	0 1 4 10 18 23 25	27	0 1 5 7 15 18 27	25	0 1 4 10 18 23 25	26	0 1 7 9 12 22 26
	28	0 1 3 8 12 22 28	30	0 1 3 7 12 20 30	30	0 1 3 7 12 20 30	26	0 1 7 9 12 22 26	28	0 1 3 8 12 22 28	26	0 1 7 9 12 22 26	27	0 1 5 7 15 18 27
	30	0 1 3 7 12 20 30					27	0 1 5 7 15 18 27			27	0 1 5 7 15 18 27	28	0 1 3 8 12 22 28
8	39	0 1 3 8 14 18 30 39	39	0 1 3 8 14 18 30 39	34	0 1 4 9 15 22 32 34	34	0 1 4 9 15 22 32 34	34	0 1 4 9 15 22 32 34	34	0 1 4 9 15 22 32 34	34	0 1 4 9 15 22 32 34
	41	0 1 3 7 15 20 31 41	41	0 1 3 7 15 20 31 41	39	0 1 3 8 14 18 30 39	39	0 1 3 8 14 18 30 39	39	0 1 3 8 14 18 30 39	39	0 1 3 8 14 18 30 39	39	0 1 3 8 14 18 30 39

Table 8 (continued)

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
9	44	1 4 10 18 20 33 40 44 45	46	3 6 7 13 22 24 36 44 49	44	2 3 7 14 27 29 37 43 46	44	2 3 7 14 27 29 37 43 46	44	0 3 9 17 19 32 39 43 44	44	0 3 9 17 19 32 39 43 44	44	0 3 9 17 19 32 39 43 44
	44	4 5 9 16 29 31 39 45 48	46	1 2 12 17 20 34 41 43 47	55	0 1 3 8 14 18 34 43 55	49	1 5 11 12 20 33 36 38 50	49	1 5 11 12 20 33 36 38 50	49	1 5 11 12 20 33 36 38 50	47	1 2 4 11 17 22 36 44 48
	57	0 1 4 6 14 21 32 48 57	47	4 5 11 16 25 33 35 48 51									49	1 5 11 12 20 33 36 38 50
			58	0 1 3 8 12 22 28 45 58										
10	55	1 3 15 22 30 33 46 50 55 56	55	2 4 16 23 31 34 47 51 56 57	55	1 2 7 11 24 27 35 42 54 56	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55
	58	2 5 10 14 28 38 39 45 58 60	55	4 6 18 25 33 36 49 53 58 59	55	6 7 12 16 29 32 40 47 59 61								
	74	0 3 5 13 22 28 29 40 60 74	77	0 1 4 6 14 23 30 41 62 77										

Table 8 (continued)

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
11	72	4 5 13 23 28 35 56 60 62 73 76	72	3 5 11 21 28 42 47 62 71 74 75	72	1 2 5 14 29 34 48 55 65 71 73	72	0 1 9 19 24 31 52 56 58 69 72	72	0 1 4 13 28 33 47 54 64 70 72	72	0 1 9 19 24 31 52 56 58 69 72	72	0 1 4 13 28 33 47 54 64 70 72
	105	0 1 3 7 12 20 30 44 65 90 105	103	1 3 4 12 17 24 34 49 53 77 104	72	3 6 17 19 23 44 51 56 66 74 75					103	1 3 4 12 17 24 34 49 53 77 104	72	0 1 9 19 24 31 52 56 58 69 72
					96	0 2 3 8 17 21 28 50 60 84 96							103	1 3 4 12 17 24 34 49 53 77 104
12	85	4 13 14 21 34 46 49 60 65 83 87 89	85	4 13 14 21 34 46 49 60 65 83 87 89	85	4 6 10 28 33 44 47 59 72 79 80 89	85	1 3 7 25 30 41 44 56 69 76 77 86	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85
	90	1 5 19 27 36 38 39 63 68 78 84 91	90	2 9 15 25 30 54 55 57 66 74 88 92	91	1 4 8 19 21 40 52 62 78 86 87 92								
					93	1 3 4 13 19 36 56 60 67 81 86 94								

Table 8 (continued)

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
13	106 110 110	10 12 15 35 47 53 69 80 95 99 108 109 116 3 4 6 26 35 45 50 61 88 95 101 109 113 1 4 11 23 27 48 65 76 78 96 105 110 111	109 109 111	6 8 9 22 32 40 60 67 79 100 104 109 115 5 7 14 17 37 52 53 77 81 95 103 108 114 2 10 13 15 29 33 50 59 74 84 106 112 113	106 111	2 4 7 27 39 45 61 72 87 91 100 101 108 7 8 11 18 30 44 46 64 73 88 105 113 118	106 106	5 12 13 22 26 41 52 68 74 86 106 109 111 2 4 7 27 39 45 61 72 87 91 100 101 108	106 111	5 12 13 22 26 41 52 68 74 86 106 109 111 1 2 4 13 29 34 49 63 71 89 102 106 112	106 111	1 8 9 18 22 37 48 64 70 82 102 105 107 3 9 21 22 26 42 52 67 74 76 103 111 114	106	0 7 8 17 21 36 47 63 69 81 101 104 106
14	229	0 1 3 7 12 20 34 44 70 86 109 180 201 229	221	1 4 18 22 29 60 62 68 84 97 149 169 181 222	206 226	15 16 22 24 37 47 75 95 109 139 144 156 185 221 0 1 3 7 12 20 34 44 70 86 109 180 201 226	169 206	0 7 15 24 34 45 57 70 84 99 115 132 150 169 2 3 5 9 17 30 50 67 86 96 126 135 157 208	206	2 3 5 9 17 30 50 67 86 96 126 135 157 208	169	0 7 15 24 34 45 57 70 84 99 115 132 150 169	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127

Table 8 (continued)

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
15	267	1 3 28 32 38 43 46 62 90 111 131 143 144 182 268	267	1 3 28 32 38 43 46 62 90 111 131 143 144 182 268	260	11 14 27 33 37 44 45 84 99 128 137 174 215 235 271	260	11 14 27 33 37 44 45 84 99 128 137 174 215 235 271	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151
16	316	5 11 15 20 45 71 78 91 99 123 126 140 253 284 303 321	316	5 11 15 20 45 71 78 91 99 123 126 140 253 284 303 321	283	3 4 7 17 36 56 79 81 87 125 142 166 192 258 265 286	283	3 4 7 17 36 56 79 81 87 125 142 166 192 258 265 286	283	3 4 7 17 36 56 79 81 87 125 142 166 192 258 265 286	283	3 4 7 17 36 56 79 81 87 125 142 166 192 258 265 286	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177
17	369	2 5 6 14 21 32 49 54 108 110 180 190 222 247 253 337 371	369	2 5 6 14 21 32 49 54 108 110 180 190 222 247 253 337 371	355	7 17 21 28 36 37 61 73 99 116 147 189 207 230 264 311 362	355	7 17 21 28 36 37 61 73 99 116 147 189 207 230 264 311 362	354	0 2 7 15 21 62 66 90 99 116 138 169 172 243 311 343 354	354	0 2 7 15 21 62 66 90 99 116 138 169 172 243 311 343 354	369	2 5 6 14 21 32 49 54 108 110 180 190 222 247 253 337 371

Table 8 (continued)

n	MBB–BC						FA		MFA					
	BB–BCM		LBB–BC		LBB–BCMLBB–BCM				FAM		LFA		LFAM	
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
18	445	0 1 3 17 29 35 71 98 102 122 147 160 212 235 256 295 338 445	436	0 6 26 30 38 39 57 73 126 128 149 240 255 265 305 319 380 436	436	0 6 26 30 38 39 57 73 126 128 149 240 255 265 305 319 380 436	463	5 9 17 19 28 35 41 70 97 98 143 146 180 246 296 301 400 468	362	14 27 36 42 54 62 93 100 130 147 149 191 202 292 306 316 375 376	445	0 1 3 17 29 35 71 98 102 122 147 160 212 235 256 295 338 445	445	0 1 3 17 29 35 71 98 102 122 147 160 212 235 256 295 338 445
19	567	0 2 18 37 43 52 92 97 130 143 150 160 172 219 356 384 387 423 567	584	3 15 16 18 34 66 109 119 133 207 243 265 271 276 355 376 478 530 587	467	3 6 25 26 51 53 58 66 104 135 139 153 243 277 319 348 402 459 470	567	0 2 18 37 43 52 92 97 130 143 150 160 172 219 356 384 387 423 567	467	3 6 25 26 51 53 58 66 104 135 139 153 243 277 319 348 402 459 470	475	3 15 16 18 34 60 66 109 119 133 207 216 243 271 276 355 376 413 478	467	3 6 25 26 51 53 58 66 104 135 139 153 243 277 319 348 402 459 470
20	673	8 10 16 26 31 48 75 95 130 171 183 264 273 364 415 444 458 565 569 681	673	8 10 16 26 31 48 75 95 130 171 183 264 273 364 415 444 458 565 569 681	649	4 23 31 43 46 71 80 81 136 150 168 181 214 298 381 467 472 483 535 653	649	4 23 31 43 46 71 80 81 136 150 168 181 214 298 381 467 472 483 535 653	615	1 2 4 9 33 50 116 126 138 154 188 197 257 294 426 477 496 517 559 616	615	1 2 4 9 33 50 116 126 138 154 188 197 257 294 426 477 496 517 559 616	578	4 8 22 27 44 47 103 110 118 131 168 180 319 354 363 364 405 432 525 582

Table 9 Near-OGRs found by BA, MBA, CSA, CSAM, FPA and FPAM

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3	3	0 1 3
4	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6	6	0 1 4 6
	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7	7	0 1 3 7			7	0 1 3 7		
5	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11	11	0 1 4 9 11
	12	0 1 3 7 12	12	0 1 3 7 12	11	0 2 7 8 11	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12
			13	0 1 4 6 13	12	0 1 3 7 12	13	0 1 4 6 13								
6	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17	17	0 1 4 10 12 17
	18	0 1 3 8 12 18	18	0 1 3 8 12 18	17	0 1 4 10 15 17	17	0 1 4 10 15 17	17	0 1 4 10 15 17	17	0 1 4 10 15 17	17	0 1 4 10 15 17	17	0 1 4 10 15 17
					17	0 1 8 11 13 17	17	0 1 8 11 13 17	17	0 1 8 11 13 17	18	0 1 3 8 12 18	18	0 1 3 8 12 18	17	0 1 8 11 13 17
					18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18					18	0 1 3 8 12 18
7	25	0 1 4 10 18 23 25	25	0 2 3 10 16 21 25	25	0 2 3 10 16 21 25	25	0 2 3 10 16 21 25	25	0 2 6 9 14 24 25	25	0 2 3 10 16 21 25	25	0 2 6 9 14 24 25	25	0 2 3 10 16 21 25
	25	0 2 3 10 16 21 25	26	0 1 7 9 12 22 26	27	0 1 5 7 15 18 27	26	0 1 7 9 12 22 26	25	0 2 3 10 16 21 25	26	0 1 7 9 12 22 26	25	0 1 4 10 18 23 25	26	0 1 7 9 12 22 26
	25	0 2 7 13 21 22 25	27	0 1 5 7 15 18 27			27	0 1 5 7 15 18 27	26	0 1 7 9 12 22 26	27	0 1 5 7 15 18 27	25	0 2 3 10 16 21 25	27	0 1 5 7 15 18 27
	28	0 1 3 8 12 22 28							27	0 1 5 7 15 18 27			26	0 1 7 9 12 22 26		
													27	0 1 5 7 15 18 27		

Table 9 (continued)

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
8	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39	34 39	0 1 4 9 15 22 32 34 0 1 3 8 14 18 30 39
9	44	0 3 9 17 19 32 39 43 44	44 57	0 3 9 17 19 32 39 43 44 0 1 4 6 14 21 32 48 57	44 58	0 3 9 17 19 32 39 43 44 0 1 3 8 12 22 28 45 58	44 47	0 3 9 17 19 32 39 43 44 1 2 4 11 17 22 36 44 48	44	0 3 9 17 19 32 39 43 44	44 47 55	0 3 9 17 19 32 39 43 44 1 2 4 11 17 22 36 44 48 0 1 3 8 14 18 34 43 55	44	0 3 9 17 19 32 39 43 44	44 47 55	0 3 9 17 19 32 39 43 44 1 2 4 11 17 22 36 44 48 0 1 3 8 14 18 34 43 55
10	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55	55	0 1 6 10 23 26 34 41 53 55

Table 9 (continued)

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
11	72	0 1 4 13 28 33 47 54 64 70 72	72	0 1 4 13 28 33 47 54 64 70 72	72	0 1 9 19 24 31 52 56 58 69 72	72 72	0 1 4 13 28 33 47 54 64 70 72 0 1 9 19 24 31 52 56 58 69 72	72	0 1 9 19 24 31 52 56 58 69 72	72 103	0 1 4 13 28 33 47 54 64 70 72 1 3 4 12 17 24 34 49 53 77 104	72	0 1 4 13 28 33 47 54 64 70 72	72 103	0 1 4 13 28 33 47 54 64 70 72 1 3 4 12 17 24 34 49 53 77 104
12	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85	85	0 2 6 24 29 40 43 55 68 75 76 85
13	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106	106	0 7 8 17 21 36 47 63 69 81 101 104 106

Table 9 (continued)

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
14	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127	127	0 5 28 38 41 49 50 68 75 92 107 121 123 127
15	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151	151	0 6 7 15 28 40 51 75 89 92 94 121 131 147 151
16	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177	177	0 1 4 11 26 32 56 68 76 115 117 134 150 163 168 177

Table 9 (continued)

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
17	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199	199	0 5 7 17 52 54 56 67 80 81 100 122 138 159 165 168 191 199
18	427	9 21 45 49 50 60 82 112 126 139 160 232 234 292 317 348 367 436	445	0 1 3 17 29 35 71 98 102 122 147 160 212 235 256 295 338 445	362	14 27 36 42 54 62 93 100 130 147 149 191 202 292 306 316 375 376	216	0 2 10 22 53 56 82 83 89 98 130 148 153 167 188 192 205 216	216	0 2 10 22 53 56 82 83 89 98 130 148 153 167 188 192 205 216	216	0 2 10 22 53 56 82 83 89 98 130 148 153 167 188 192 205 216	216	0 2 10 22 53 56 82 83 89 98 130 148 153 167 188 192 205 216	216	0 2 10 22 53 56 82 83 89 98 130 148 153 167 188 192 205 216
19	467	3 6 25 26 51 53 58 66 104 135 139 153 243 277 319 348 402 459 470	467	3 6 25 26 51 53 58 66 104 135 139 153 243 277 319 348 402 459 470	475	3 15 16 18 34 60 66 109 119 133 207 216 243 271 276 355 376 413 478	246	0 4 13 15 42 56 59 77 93 116 126 138 146 174 214 221 240 245 246	246	0 4 13 15 42 56 59 77 93 116 126 138 146 174 214 221 240 245 246	246	0 4 13 15 42 56 59 77 93 116 126 138 146 174 214 221 240 245 246	246	0 4 13 15 42 56 59 77 93 116 126 138 146 174 214 221 240 245 246	246	0 4 13 15 42 56 59 77 93 116 126 138 146 174 214 221 240 245 246

Table 9 (continued)

n	BA		MBA		CSA						CSAM		FPA		FPAM	
					BAM		LBA		LBAM							
	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks	Length	Position of marks
20	578	4 8 22 27	578	4 8 22	578	4 8 22 27	283	0 24 30	283	0 24 30	283	0 24 30	283	0 24 30	283	0 24 30 43
	615	44 47		27 44	593	44 47		43 55		43 55		43 55		43 55		55 71 75
		103 110		47 103		103 110		71 75		71 75		71 75		71 75		89 104
		118 131		110 118		118 131		89 104		89 104		89 104		89 104		125 127
		168 180		131 168		168 180		125 127		125 127		125 127		125 127		162 167
		319 354		180 319		319 354		162 167		162 167		162 167		162 167		189 206
		363 364		354 363		363 364		189 206		189 206		189 206		189 206		215 272
		405 432		364 405		405 432		215 272		215 272		215 272		215 272		275 282
		525 582		432 525		525 582		275 282		275 282		275 282		275 282		283
		1 2 4 9 33		582		14 20 24		283		283		283		283		
		50 116				58 89										
		126 138				94 112										
		154 188				115 170										
		197 257				197 221										
		294 426				222 230										
		477 496				237 381										
		517 559				449 505										
		616				546 568										
						607										

References

Afshar MH, Motaei I (2011) Constrained big bang-big crunch algorithm for optimal solution of large scale reservoir operation problem. Int J Optim Civil Eng 2:357–375

Aggarwal GP (2001) Nonlinear fiber optics, 2nd edn. Academic Press, San Diego, CA

Atkinson MD, Santoro N, Urrutia J (1986) Integer sets with distinct sums and differences and carrier frequency assignments for nonlinear repeaters. IEEE Trans Commun 34(6):614–617

Ayari N, Thé Van Luong, Jemai A (2010) A hybrid genetic algorithm for Golomb ruler problem. In: ACS/IEEE international conference on computer systems and applications (AICCSA-2010), pp 1–4

Babcock WC (1953) Intermodulation interference in radio systems. Bell Syst Tech J 32:63–73

Bansal S (2014) Optimal Golomb ruler sequence generation for FWM crosstalk elimination: soft computing versus conventional approaches. Appl Soft Comput 22:443–457

Bansal S (2017) Nature-inspired based multi-objective hybrid algorithms to find near-OGRs for Optical WDM systems and their comparison. In: Hamou RM (ed) Advanced metaheuristic methods in big data retrieval and analytics. IGI Global Publisher, Philadelphia, pp 175–211

Bansal S (2019) A comparative study of nature-inspired metaheuristic algorithms in search of near-to-optimal Golomb rulers for the FWM crosstalk elimination in WDM systems. Appl Artif Intell 33(14):1199–1265

Bansal S, Sharma K (2017) Nature-inspired based modified multi-objective BB-BC algorithm to find near-OGRs for optical WDM systems and its performance comparison. In: Hamou RM (ed) Handbook of research on biomimicry in information retrieval and knowledge management. IGI Global Publisher, Philadelphia, pp 1–25

Bloom GS, Golomb SW (1977) Applications of numbered undirected graphs. Proc IEEE 65(4):562–570

Blum EJ, Biraud F, Ribes JC (1974) On optimal synthetic linear arrays with applications to radio astronomy. IEEE Trans Antennas Propag 22:108–109

Colannino J (2003) Circular and modular Golomb rulers. http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2003/JustinColannino/

Cotta C, Fernández AJ (2005) Analyzing fitness landscapes for the optimal Golomb ruler problem. In: Gottlieb J, Raidl G (eds) Evolutionary computation in combinatorial optimization. Lecture notes in computer science, vol 3448. Springer, Berlin, pp 68–79

Cotta C, Hemert JV (eds) (2008) Recent advances in evolutionary computation for combinatorial optimization. In: Studies in computational intelligence, vol 153. Springer

Cotta C, Dotú I, Fernández AJ, Hentenryck PV (2006) A memetic approach to Golomb rulers. In: Parallel problem solving from nature—PPSN IX, Lecture notes in computer science, vol 4193, pp 252–261, Springer, Berlin

Dimitromanolakis A (2002) Analysis of the Golomb ruler and the sidon set problems, and determination of large, near-optimal Golomb rulers. Master's Thesis, Department of Electronic and Computer Engineering, Technical University of Crete

Distributed.net. Project OGR. http://www.distributed.net/ogr

Dollas A, Rankin WT, McCracken D (1998) A new algorithm for golomb ruler derivation and proof of the 19 mark ruler. IEEE Trans Inf Theory 44(1):379–382

Dotú I, Hentenryck PV (2005) A simple hybrid evolutionary algorithm for finding Golomb rulers. In: The 2005 IEEE congress on evolutionary computation, vol 3, pp 2018–2023

Drakakis K (2009) A review of the available construction methods for Golomb rulers. Adv Math Commun 3(3):235–250

Drakakis K, Rickard S (2010) On the construction of nearly optimal Golomb rulers by unwrapping costas arrays. Contemp Eng Sci 3(7):295–309

Erol OK, Eksin I (2006) A new optimization method: big bang-big crunch. Adv Eng Softw 37:106–111

Fang RJF, Sandrin WA (1977) Carrier frequency assignment for non-linear repeaters. Comsat Techn Rev 7:227–245

Forghieri F, Tkach RW, Chraplyvy AR (1995) WDM systems with unequally spaced channels. J Lightw Technol 13:889–897

Galinier P, Jaumard B, Morales R, Pesant G (2001) A constraint-based approach to the Golomb ruler problem. In: 3rd international workshop on integration of AI and OR techniques (CP-AI-OR 2001)

Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput Int J Simul Based Eng 29(1):17–35

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Guo S-S, Wang J-S, Ma X-X (2019) Improved bat algorithm based on multipopulation strategy of Island model for solving global function optimization problem. Comput Intell Neurosci 2019(1):1–12

http://theinf1.informatik.uni-jena.de/teaching/ss10/oberseminar-ss10

http://mathworld.wolfram.com/PerfectRuler.html

http://mathworld.wolfram.com/GolombRuler.html

Hwang B, Tonguz OK (1998) A generalized suboptimum unequally spaced channel allocation technique—part I. In IM/DDWDM systems. IEEE Trans Commun 46:1027–1037

Iglesias A, Gálvez A, Suárez P, Shinya M, Yoshida N, Otero C, Manchado C, Gomez-Jauregui V (2018) Cuckoo search algorithm with Lévy flights for global-support parametric surface approximation in reverse engineering. Symmetry 10(58):1–25

Koziel S, Yang X-S (eds) (2011) Computational optimization, methods and algorithms. In: Studies in computational intelligence, vol 356. Springer

Kwong WC, Yang GC (1997) An algebraic approach to the unequal-spaced channel–allocation problem in WDM lightwave systems. IEEE Trans Commun 45(3):352–359

Lam AW, Sarwate DV (1988) On optimal time-hopping patterns. IEEE Trans Commun 36:380–382

Lavoie P, Haccoun D, Savaria Y (1991) New VLSI architectures for fast soft-decision threshold decoders. IEEE Trans Commun 39(2):200–207

Leitao T (2004) Evolving the maximum segment length of a Golomb ruler. In: Genetic and evolutionary computation conference, USA

Li Y, Li X, Liu J, Ruan X (2019) An improved bat algorithm based on Lévy flights and adjustment factors. Symmetry 11(7):1–19

Mareli M, Twala B (2018) An adaptive Cuckoo search algorithm for optimisation. Appl Comput Inform 14(2):107–115

Memarsadegh N (2013) Golomb patterns: introduction, applications, and citizen science game. In: Information Science and Technology (IS&T), Seminar Series NASA GSFC. http://istcolloq.gsfc.nasa.gov/fall2013/presentations/memarsadeghi.pdf

Mitchell M (2004) An introduction to genetic algorithms. Prentice Hall of India Pvt. Ltd., New Delhi

Price K, Storn R, Lampinen J (2005) Differential evolution—a practical approach to global optimization. Springer, Berlin

Project Educational NASA Computational and Scientific Studies (enCOMPASS). http://encompass.gsfc.nasa.gov/cases.html

Rajasekaran S, Vijayalakshmi Pai GA (2004) Neural networks, fuzzy logic, and genetic algorithms-synthesis and applications. Prentice Hall of India Pvt. Ltd., New Delhi

Randhawa R, Sohal JS, Kaler RS (2009) Optimum algorithm for WDM channel allocation for reducing four-wave mixing effects. Optik 120(2009):898–904

Rankin WT (1993) Optimal Golomb rulers: an exhaustive parallel search implementation. M.S. Thesis, Duke University. http://people.ee.duke.edu/~wrankin/golomb/golomb.html. Accessed Dec 2003

Robinson JP (1979) Optimum Golomb rulers. IEEE Trans Comput 28(12):183–184

Robinson JP (2000) Genetic search for Golomb arrays. IEEE Trans Inf Theory 46(3):1170–1173

Robinson JP, Bernstein AJ (1967) A class of binary recurrent codes with limited error propagation. IEEE Trans Inf Theory IT-13:106–113

Sardesai HP (1999) A simple channel plan to reduce effects of nonlinearities in dense WDM systems. Lasers Electro-Opt 5:183–184

Shearer JB (1990) Some new optimum Golomb rulers. IEEE Trans Inf Theory 36:183–184

Shearer JB (1998) Some new disjoint Golomb rulers. IEEE Trans Inf Theory 44(7):3151–3153

Singh K, Bansal S (2013) Suppression of FWM crosstalk on WDM systems using unequally spaced channel algorithms—a survey. Int J Adv Comput Sci Softw Eng (IJARCSSE) 3(12):25–31

Soliday SW, Homaifar A, Lebby GL (1995) Genetic algorithm approach to the search for Golomb rulers. In: Proceedings of the sixth international conference on genetic algorithms (ICGA-95). Morgan Kaufmann, pp 528–535

Storn R, Price KV (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

Tabakov PY (2011) Big bang-big crunch optimization method in optimum design of complex composite laminates. World Acad Sci Eng Technol 77:835–839

Thing VLL, Shum P, Rao MK (2004) Bandwidth-efficient WDM channel allocation for four-wave mixing-effect minimization. IEEE Trans Commun 52(12):2184–2189

Tonguz OK, Hwang B (1998) A generalized suboptimum unequally spaced channel allocation technique—part II: in coherent WDM systems. IEEE Trans Commun 46:1186–1193

Yang X-S (2010a) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, Bristol

Yang X-S (2010b) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84
Yang X-S (2010c) Firefly algorithm, Levy flights and global optimization. In: Bramer M, Ellis R, Petridis (eds) Research and development in intelligent systems XXVI. Springer, London, pp 209–218
Yang X-S (2010d) A new metaheuristic bat–inspired algorithm. In: Gonzalez JR et al (eds) Nature inspired coop-erative strategies for optimization (NISCO-2010), Studies in computational intelligence, vol 284. Springer, Berlin, pp 65–74
Yang X-S (2011a) Chaos-enhanced firefly algorithm with automatic parameter tuning. Int J Swarm Intell Res (IJSIR) 2(4):1–11
Yang X-S (2011b) Review of metaheuristics and generalized evolutionary walk algorithm. Int J Bio-Inspired Comput 3(2):77–84
Yang X-S (2012a) Nature-inspired mateheuristic algorithms: success and new challenges. J Comput Eng Inf Technol (JCEIT) 1(1):1–3
Yang X-S (2012b) Flower pollination algorithm for global optimization. In: Unconventional computation and natural computation 2012, Lecture notes in computer science, vol 7445, Springer, Berlin, pp 240–249
Yang X-S (2013a) Optimization and metaheuristic algorithms in engineering. In: Yang XS, Gandomi AH, Talatahari S, Alavi AH (eds) Metaheursitics in water, geotechnical and transport engineering. Elsevier, New York. https://doi.org/10.1016/B978-0-12-398296-4.00001-5
Yang X-S (2013b) Bat algorithm: literature review and applications. Int J Bio-Inspired Computat 5(3):141–149
Yang X-S, Deb S (2010a) Eagle strategy using levy walk and firefly algorithms for stochastic optimization. In: Gonzalez JR et al (eds) Nature inspired cooperative strategies for optimization (NISCO-2010), Studies in computational intelligence, vol 284. Springer, Berlin, pp 101–111
Yang X-S, Deb S (2010b) Engineering optimisation by Cuckoo search. Int J Math Model Numer Optim 1(4):330–343
Yang X-S, Deb S (2014) Cuckoo search: recent advances and applications. Neural Comput Appl 24(1):169–174
Yang X-S, He XS (2013) Firefly algorithm: recent advances and applications. Int J Swarm Intell 1(1):36–50
Yang X-S, Karamanoglu M, He XS (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237
Yesil E, Urbas L (2010) Big bang-big crunch learning method for fuzzy cognitive maps. World Acad Sci Eng Technol 71:815–824