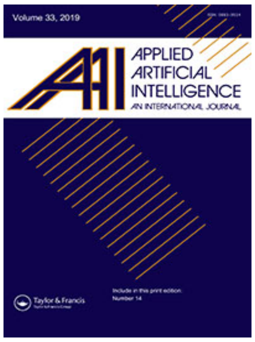

Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

A Comparative Study of Nature-Inspired Metaheuristic Algorithms in Search of Near-to-optimal Golomb Rulers for the FWM Crosstalk Elimination in WDM Systems

Shonak Bansal

APPLIED ARTIFICIAL INTELLIGENCE
2019, VOL. 33, NO. 14, 1199–1265
https://doi.org/10.1080/08839514.2019.1683977

A Comparative Study of Nature-Inspired Metaheuristic Algorithms in Search of Near-to-optimal Golomb Rulers for the FWM Crosstalk Elimination in WDM Systems

Shonak Bansal

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

ABSTRACT
Nowadays, nature-inspired metaheuristic algorithms are the most powerful optimizing algorithms for solving NP-complete problems. This paper proposes five recent approaches to find near-optimal Golomb ruler (OGR) sequences based on nature-inspired algorithms in a reasonable time. The optimal Golomb ruler sequences found their application in channel-allocation method that allows suppression of the crosstalk due to four-wave mixing (FWM) in optical wavelength division multiplexing (WDM) systems. The simulation results conclude that the proposed nature-inspired metaheuristic optimization algorithms are superior to the existing conventional computing algorithms, i.e., Extended Quadratic Congruence (EQC) and Search algorithm (SA) and nature-inspired algorithms, i.e., Genetic algorithms (GAs), Biogeography-based optimization (BBO) and simple Big bang–Big crunch (BB-BC) optimization algorithm to find near-OGRs in terms of ruler length, total optical channel bandwidth and computation time.

Introduction

There exists a rich collection of nonlinear optical effects (Aggarwal, 2001; Babcock 1953; Chraplyvy 1990; Forghieri et al. 1994; Kwong and Yang 1997; Saaid 2010; Singh and Bansal 2013; Sugumaran et al. 2013; Thing, Shum, and Rao 2004) in optical WDM systems, each of which manifests itself in a unique way. Out of these nonlinearities the FWM 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

CONTACT Shonak Bansal shonakk@gmail.com Electronics and Communication Engineering Department, PEC University of Technology, Chandigarh, India

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 (Aggarwal, 2001; Babcock 1953; Chraplyvy 1990; Forghieri et al. 1994; Kwong and Yang 1997; Saaid 2010; Singh and Bansal 2013; Sugumaran et al. 2013; Thing, Shum, and Rao 2004).

To suppress the crosstalk due to FWM signals in optical WDM systems, several unequally spaced channel allocation (USCA) algorithms have been proposed in the literatures (Atkinson, Santoro, and Urrutia 1986; Compunity 0000; Forghieri, Tkach, and Chraplyvy 1995; Hwang and Tonguz 1998; Kwong and Yang 1997; Randhawa, Sohal, and Kaler 2009; Sardesai 1999; Tonguz and Hwang 1998). However, the algorithms (Atkinson, Santoro, and Urrutia 1986; Compunity 0000; Forghieri, Tkach, and Chraplyvy 1995; Hwang and Tonguz 1998; Kwong and Yang 1997; Randhawa, Sohal, and Kaler 2009; Sardesai 1999; Tonguz and Hwang 1998) have the drawback of increased optical 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-OGR sequences (Babcock 1953; Bloom and Golomb 1977; Shearer 1998; Thing, Rao, and Shum 2003) to suppress FWM crosstalk in optical WDM systems.

Studies have been shown that Golomb rulers represent a class of NP-complete (oberseminar 0000) problems. For higher-order marks, the exhaustive computer search (Robinson 1979; Shearer 1990) of such problems is difficult. In literatures (Cotta et al. 2006; Galinier et al. 2001; Leitao 2004; Rankin 1993; Robinson 1979; Shearer 1990), there are numerous algorithms 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 Memetic approach (Cotta et al. 2006), Tabu search (TS) (Cotta et al. 2006), GAs (Ayari, Luong, and Jemai 2010; Bansal 2014; Robinson 2000; Soliday, Homaifar, and Lebby 1995) and its hybridizations (HGA) (Ayari, Luong, and Jemai 2010), BBO (Bansal 2014, 2011; Bansal et al. 2011), BB-BC (Bansal, Kumar, and Bhalla 2013) and hybrid evolutionary (HE) algorithms (Dotú and Hentenryck 2005) 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 recent nature-inspired algorithms, namely, Big bang–Big crunch (BB-BC) optimization algorithm, Firefly algorithm (FA), Bat algorithm (BA), Cuckoo search algorithm (CSA), Flower pollination algorithm (FPA) and their modified form 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.

The structure of this paper is organized as follows: Section II presents the concept of Golomb rulers. Section III describes a brief account of nature-inspired based optimization algorithms. Section IV presents the problem formulation. Section V provides experimental results comparing with conventional and nature-inspired based optimization algorithms of finding unequal channel spacing (UCS). Conclusions and future work are outlined in Section VI.

Golomb Rulers

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

Golomb rulers are an ordered set of unevenly marks at non-negative integer locations such that no distinct pairs of numbers from the set have the same difference (Cotta et al. 2007; Distributed.net 0000; Drakakis 2009; Drakakis and Rickard 2010). These numbers are referred to as *marks*. The difference between the largest and smallest number is referred to as the *ruler length*. The number of marks on a ruler is referred to as the *ruler size* (Bansal 2014; Bansal et al. 2011; Bloom and Golomb 1977).

An OGR is the shortest length ruler for a given number of marks (GolombRuler 0000; Perfect ruler 0000). There can be multiple different OGRs for a specific number of marks. However, the unique optimal Golomb 4-marks ruler is shown in Figure 1, which measures all the distances from 0 to 6.

A perfect Golomb ruler measures all the integer distances from 0 to ruler length (oberseminar 0000; Soliday, Homaifar, and Lebby 1995). The ruler length RL of an n-mark perfect Golomb ruler is given by equation (1) (Rankin 1993):

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

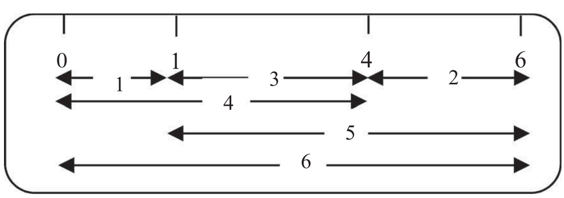

Figure 1. A 4-marks OGR with its associated distances.

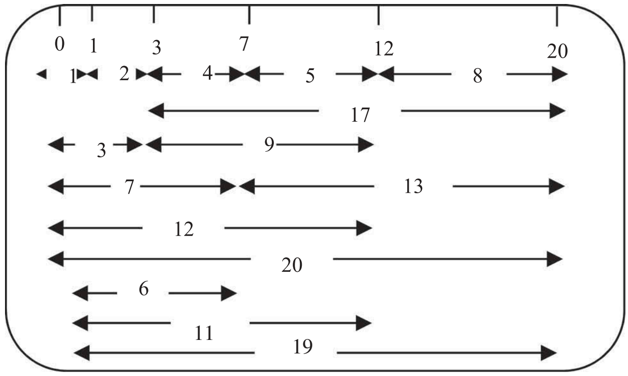

Figure 2. A 6-marks non-OGR with its associated distances.

For example, the set (0, 1, 3, 7, 12, 20), shown in Figure 2 is a non-optimal 6-marks Golomb ruler having length 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 sequence. The distance associated between each pair of marks is also shown in Figure 2.

The OGRs play an important role in a variety of real-world applications including radio frequency allocation, sensor placement in X-ray crystallography, computer communication network, pulse phase modulation, circuit layout, geographical mapping, self-orthogonal codes, VLSI architecture, 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; Blum, Biraud, and Ribes 1974; Cotta and Fernández 2005; Dimitromanolakis 2002; Dollas, Rankin, and McCracken 1998; Fang and Sandrin 1977; Lam and Sarwate 1988; Lavoie, Haccoun, and Savaria 1991; Memarsadegh 2013; Project Educational NASA Computational and Scientific Studies (enCOMPASS) 0000; Rankin 1993; Robinson and Bernstein 1967; Soliday, Homaifar, and Lebby 1995).

On applying OGRs to the channel allocation, it was possible to achieve the smallest distinct number to be used for the 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.

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 the 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 0000; Goldberg 1989; Koziel and Yang 2011; Mitchell 2004; Rajasekaran and Vijayalakshmi Pai 2004; Yang 2013a, 2010a, 2012a).

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 subdomain. 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 nature-inspired metaheuristic algorithms, which is the main technical contribution of this paper. In 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 equation (2):

$$MR_i^t = \frac{f_i^t}{Max(f^t)} \tag{2}$$

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 (Price, Storn, and Lampinen 2005; Storn and Price 1997) use throughout this paper is given by the equation (3):

$$x_i^t = x_i^{t-1} + p_m(x_{best}^{t-1} - x_i^{t-1}) + p_m(x_{r_1}^{t-1} - x_{r_2}^{t-1}) \tag{3}$$

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 the equation (4):

$$L(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s >> s_0 > 0) \tag{4}$$

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 nature-inspired 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 (Ayari, Luong, and Jemai 2010; Bansal 2014, 2011; Bansal, Kumar, and Bhalla 2013; Bansal et al. 2011; Cotta et al. 2006; Dotú and Hentenryck 2005; Robinson 2000; Soliday, Homaifar, and Lebby 1995) to find near-OGRs.

A. *Big Bang–Big Crunch Optimization Algorithm and Its Modified Forms*

Erol et al. (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 (Afshar and Motaei 2011; Erol and Eksin 2006; Genc, Eksin, and Erol 2013; Kripka and Kripka 2008; Kumbasar et al. 2008; Tabakov 2011; Yesil and Urbas 2010). In BB-BC, the center of mass mathematically is computed by the equation (5) (Erol and Eksin 2006):

$$x_c = \frac{\sum_{i=1}^{Popsize} \frac{1}{f_i} x_i}{\sum_{i=1}^{Popsize} \frac{1}{f_i}} \tag{5}$$

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 center 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 equation (6) (Erol and Eksin 2006):

$$x_{new} = x_c + r \times c_1 \times \frac{(x_{\max} - x_{\min})}{1 + t/c_2} \tag{6}$$

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, equation (6) is randomized via Lévy flights as given by equation (7).

$$x_{new} = x_c + r \times c_1 \times \frac{(x_{\max} - x_{\min})}{1 + t/c_2} \oplus L(\lambda) \tag{7}$$

The product $\oplus$ means entrywise multiplications and $L(\lambda)$ is the Lévy flight-based step size given mathematically by equation (4).

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

```
Modified Big Bang–Big Crunch (MBB–BC) Algorithm
Begin
  /* Big Bang Phase */
       Generate a random set of candidates (population);
  /* End of Big Bang Phase */
  While not T                          /* T is a termination criterion */
    Compute the fitness value of all the candidate solutions;
    Sort the population from best to worst based on fitness (cost)
    value;
    /* Big Crunch Phase */
        Compute the center of mass;
    /* End of Big Crunch Phase */
    Calculate new candidates around the center of mass by adding
     or subtracting a normal random number whose value decreases
     as the iterations elapse via Lévy flight;    /* Big Bang Phase */
    /* Mutation */
        Compute mutation rate probability MR_i via equation (2);
        If ( MR_i < rand)
          Perform mutation via equation (3);
        End if
    /* End of mutation */
    Rank the candidates and find the current best;
  End while
  Postprocess results and visualization;
End
```

Figure 3. General pseudo-code for MBB-BC Algorithm.

B. *Firefly Algorithm and Its Modified Forms*

X. S. Yang (Koziel and Yang 2011; Yang 2013a, 2010a, 2012a, 2009) 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 location X can simply be proportional to the objective function, i.e., $I(X) \propto f(X)$ (Koziel and Yang 2011; Yang 2013a, 2010a, 2012a, 2009, 2010b, 2010c, 2011a, 2011b; 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 2009). That is

$$I = I_0 e^{-\gamma r} \tag{8}$$

where I_0 is 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 equation (9) (Yang 2009):

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{9}$$

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 equation (10) (Yang 2009):

$$r_{ij} = \|X_i - X_j\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{10}$$

where $x_{i,k}$ is the kth component of the spatial coordinate X_i of ith firefly and d is the number of dimensions in search space. The movement of a firefly i is attracted to another more brighter firefly j is determined by equation (11) (Yang 2009):

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}\left(X_j - X_i\right) + \alpha(rand - 0.5) \tag{11}$$

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 equations (2) and (3) are 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 equation (11) is randomized via Lévy flights (equation (4)). The firefly movement equation for LFA is given by equation (12):

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}\left(X_j - X_i\right) + \alpha.sign(rand - 0.5) \oplus L(\lambda) \tag{12}$$

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 Figure 4. If lines 15 to 20 in Figure 4 are removed and in line 13 Lévy flight distributions are not used then Figure 4 corresponds to the general pseudo-code for simple FA. If Lévy flight distributions in line 13 are not used in Figure 4, then it corresponds to the general pseudo-code for FAM and if no modifications in Figure 4 are performed then it represents the general pseudo-code for LFAM algorithm.

C. Bat Algorithm and Its Modified Forms

X. S. Yang (Koziel and Yang 2011; Yang 2013a, 2010a, 2012a, 2011b, 2010d), 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 2010d) uses the following three idealized rules:

```
1. Modified Firefly Algorithm (MFA)
2.  Begin
3.    /* MFA parameter initialization */
4.        Define objective function f(X);  X = (x_1,..., x_d)^T;
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 (2);
17.            If (MR_i < rand)
18.               Perform mutation via equation (3);
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
```

Figure 4. General pseudo-code for MFA.

(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 $A°$.

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 the following equations (Cotta and Hemert 0000; Yang 2013a, 2010d, 2011c, 2013b; Yang and Gandomi 2012):

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{13}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i \tag{14}$$

$$x_i^t = x_i^{t-1} + v_i^{t-1} \tag{15}$$

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 equation (16):

$$x_{new} = x_{best} + \varepsilon A^t \tag{16}$$

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 equations (17) and (18), respectively, as iterations proceed:

$$A_i^t = \alpha A_i^{t-1} \tag{17}$$

$$r_i^t = r_i^0[1 - e^{-\gamma t}] \tag{18}$$

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 2010d, 2011c, 2013b; Yang and Gandomi 2012).

By combining the characteristics of mutation strategy (equations (2) and (3)), Lévy flights distribution (equation (4)), 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 equation (16) is given by equation (19) as:

$$x_{new} = x_{best} + \varepsilon A^t \oplus L(\lambda) \tag{19}$$

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

D. *Cuckoo Search Algorithm and Its Modified Form*

X. S. Yang et al. (Gandomi, Yang, and Alavi 2013; Yang and Deb 2010b, 2009, 2014), inspired by brood parasitism of some cuckoo species, developed a novel 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 new algorithm, X. S. Yang et al. use the following three idealized rules:

```
1. Modified Bat Algorithm (MBA)
2.  Begin
3.    /* MBA parameter initialization */
4.       Define objective function f(x);  x = (x1,..., xd)^T;
5.       Generate initial bat population xi and initial velocities vi (i = 1,
6.       2,...n);
7.       Define pulse frequency fi at xi;
8.       Initialize pulse rates ri and the loudness Ai;
9.    /* End of MBA parameter initialization */
10.   While not t                              /* t is a termination criterion */
11.      Generate new solutions by adjusting frequency, and updating
12.      velocities and locations/solutions by Lévy flights;
13.      If (rand > ri)
14.          Select a solution among the best solutions;
15.          Generate a local solution around the selected best solution;
16.      Else
17.        /* Mutation */
18.            Compute mutation rate probability MRi via equation (2);
19.            If (MRi < rand)
20.               Perform mutation via equation (3);
21.            End if
22.        /* End of Mutation */
23.      End if
24.      Generate a new solution by flying randomly;
25.      If (rand < ri and f(xi) < f(x*))
26.           Accept the new solutions;
27.           Increase ri and reduce Ai;
28.      End if
29.      Rank the bats and find the current best;
30.   End while
31.   Postprocess results and visualization;
32. End
```

Figure 5. General pseudo-code for MBA.

(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;
(3) The number of available host nests is fixed, and a host can discover an alien egg with probability $p_a \epsilon$ [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 given by equation (20) (Yang and Deb 2010b):

$$x_i^{\ t} = x_i^{\ t-1} + \alpha \oplus L(\lambda) \tag{20}$$

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 Figure 6. If the concept of mutation (lines 9 to 14) is withdrawn from Figure 6, then it corresponds to general pseudo-code for simple CSA.

E. *Flower Pollination Algorithm and Its Modified Form*

X. S. Yang et al. (Yang 2012b; Yang, Karamanoglu, and He 2013, 2014), inspired by the flow pollination process of flowering plants, introduced a novel nature-inspired optimization algorithm called Flower pollination algorithm (FPA). For describing this novel metaheuristic algorithm, the authors in (Yang 2012b) use the following four idealized rules:

(1) For global pollination process, biotic cross-pollination is used and pollen-carrying pollinators obey Lévy flights movements.
(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.

```
Cuckoo Search Algorithm with Mutation (CSAM)
Begin
  /* CSAM parameter initialization */
     Define objective function f(x);   x = (x1,…, xd)^T;
     Generate initial population of n host nests xi (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 MRi via equation (2);
        If (MRi < rand)
           Perform mutation via equation (3);
        End if
     /* End of mutation */
     Evaluate its quality/fitness Fi;
     Choose a nest among n (say j) randomly;
     If (Fi > Fj),
        Replace j by the new solution;
     End if
     A fraction (pa) 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
```

Figure 6. General pseudo-code for CSAM.

(4) The interaction of local pollination and global pollination can be controlled by a switch probability $p \in [0,1]$, with a slight bias toward 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 (21) (Yang 2012b):

$$x_i^t = x_i^{t-1} + \gamma L(\lambda)(x_* - x_i^{t-1}) \tag{21}$$

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 (22) (Yang 2012b):

$$x_i^t = x_i^{t-1} + \in (x_j^{t-1} - x_k^{t-1}) \tag{22}$$

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 (equations (2) and (3)) 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 Figure 7. The only difference in the pseudo–code for FPA and FPAM is only the addition of mutation

```
Flower Pollination Algorithm with Mutation (FPAM)
Begin
 /* FPAM parameter initialization */
   Define objective function f(x);  x = (x1,..., xd)^T;
   Initialize a population of n flowers/pollen gametes with random
   solutions;
   Find the best solution g* in the initial population;
   Define a switch probability p ∈ [0,1] ;
 /* End of FPAM parameter initialization */
 While not t                              /* t is a termination criterion */
    For i = 1 : n                                      /*all n flowers */
      If (rand < p)
          Draw a (d–dimensional) step vector L via Lévy flights;
          Perform global pollination via equation (21);
      Else
          Draw ∈ from a uniform distribution in [0,1];
          Perform local pollination via equation (22);
      End if
      /* Mutation */
         Compute mutation rate probability MRi via equation (2);
         If (MRi < rand)
            Perform mutation via equation (3);
         End if
      /* End of mutation */
      Evaluate new solutions;
      If new solutions are better, update them in the population;
    End for
    Rank the solutions and Find the current best solution x* ;
 End while
 Postprocess results and visualization;
End
```

Figure 7. General pseudo–code for FPAM.

(lines 19 to 24) in Figure 7. If lines 19 to 24 are not used in Figure 7 then it corresponds to the general pseudo–code for simple FPA.

Finding Near-OGRS: Problem Formulation

Both simplicity and efficiency attracts researchers toward natural phenomenon to solve NP-complete and complex optimization problems. The first problem investigated in this research is to find Golomb ruler sequences for unequal channel allocation. Second problem is to obtain either optimal or near-optimal Golomb rulers 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 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 given by the equations (23) and (24), respectively, as:

Ruler Length (*RL*):

$$RL = \sum_{i=1}^{n-1} (CS)_i \tag{23a}$$

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

Alternatively, equation (23a) can also be rewritten as:

$$RL = IE(n) - IE(1) \tag{23b}$$

Total Bandwidth (*TBW*):

$$TBW = \sum_{i=1}^{n} (IE)_i \tag{24}$$

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

where $i, j = 1, 2, \ldots, n$ with $i \neq j$ are distinct in both equations (23) and (24).

A. *Nature-Inspired Algorithms to Find Near-OGRs*

The general pseudo-code to find near-OGR sequences by using nature-inspired optimization algorithms proposed in this paper is shown in Figure 8. The core of the proposed nature-inspired algorithms is lines 19 to 30 which find Golomb ruler sequences 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

```
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
```

Figure 8. General pseudo–code in search of near-to-optimal Golomb rulers by using nature–inspired metaheuristic algorithms.

size (*Popsize*). Since there are many solutions, a replacement strategy must be performed as shown in Figure 8 to remove the worst individuals. This means that the proposed algorithms maintains a fixed population of rulers and performs a fixed number of iterations until either an optimal or near-to-optimal solution is found.

Simulation Results

This section presents the performance of proposed nature-inspired optimization algorithms and their performance comparison with best known OGRs (Bloom and Golomb 1977; Colannino 2003; Dollas, Rankin, and McCracken 1998; GolombRuler 0000; Rankin 1993; Shearer 1990, 2001, 0000), two of the conventional computing algorithms, i.e., EQC and SA (Kwong and Yang 1997; Randhawa, Sohal, and Kaler 2009) and three existing nature-inspired algorithms, i.e., GAs (Bansal 2014), BBO (Bansal 2014, 2011; Bansal et al. 2011) and simple BB-BC (Bansal, Kumar, and Bhalla 2013), of finding unequal channel spacing. All the proposed algorithms to find near-OGRs have coded and tested in Matlab-7.6.0 (R2008a) (Pratap 2010) 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 3Gb and hard drive of 320Gb.

A. *Simulation 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 nature-inspired algorithms have finally been settled as shown in Table 1 to 5. The selection of a suitable parameter values for nature-inspired algorithms are problem specific as there are no concrete rules.

B. *Near–OGR Sequences*

With the above-mentioned parameters setting (Tables 1–5), 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. A set of 10 trials with n = 6, 8 and 15 for all the proposed nature-inspired algorithms are reported in Tables 6–9. The performance of all the sets is nearly the same as reported in Tables 6–9. The generated near-OGR sequences for different values of marks by proposed nature-inspired algorithms are shown in Appendix–A. It has been verified that all the generated sequences are Golomb rulers. Although the proposed algorithms find same near-OGR sequences, but the difference is in required maximum number of iterations and computational time which is discussed in the following subsections.

C. *Influence of Selecting Different Population Size on the Performance of Proposed Algorithms*

In this subsection, the influence of selecting different population size (*Popsize*) on the performance of proposed nature-inspired 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 (Artifical Intelligence 0000). Table 10 shows the influence of *Popsize* on the performance of MBB-BC, FA and MFA, while Table 11 shows the influence of *Popsize* on the performance of BA, MBA, CSA, CSAM, FPA and FPAM algorithms to find near-OGRs for 7, 9 and 14-marks. In this experiment, all the parameter settings for proposed nature-inspired algorithms are same as mentioned in Tables 1–5.

Golomb ruler sequences realized from 10 to 16-marks by Tabu search algorithm (Cotta et al. 2006), maximum *Popsize* set was 190. The hybrid approach proposed in (Ayari, Luong, and Jemai 2010) to find Golomb rulers

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
A	0.5
B	0.2
Γ	1.0
p_m	0.05

Table 3. Simulation parameters for BA and MBA.

Parameter	Value
A°	0.8
r°	0.5
p_m	0.01

Table 4. Simulation parameters for CSA and CSAM.

Parameter	Value
A	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

from 11 to 23-marks, the *Popsize* was set between 20 and 2000. The near-OGR sequences found by algorithms GAs and BBO (Bansal 2014), maximum *Popsize* set was 30. For the hybrid evolutionary (HE) algorithms (Dotú and Hentenryck 2005), to find near-OGRs, maximum *Popsize* set was 50.

From Tables 10 and 11, it is noted that *Popsize* has little significant effect on the performance of all proposed nature-inspired optimization algorithms. By carefully looking at the results, the *Popsize* of 10 in all proposed algorithms was found to be adequate for finding near-OGR sequences.

Table 6. Performance of Proposed MBB-BC Algorithms to find Near-OGRs for different channels in a set of 10 trials.

	BB–BCM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0592	39	113	0.1984	267	1322	1.174e+03
2	18	42	0.0589	41	118	0.1988	267	1322	1.194e+03
3	17	44	0.0577	39	113	0.1989	267	1322	1.191e+03
4	18	42	0.0590	39	113	0.1984	267	1322	1.187e+03
5	17	44	0.0588	39	113	0.1989	267	1322	1.188e+03
6	17	44	0.0587	41	118	0.1986	267	1322	1.187e+03
7	17	44	0.0589	41	118	0.1988	267	1322	1.178e+03
8	18	42	0.0612	39	113	0.1982	267	1322	1.189e+03
9	17	44	0.0586	39	113	0.1985	267	1322	1.191e+03
10	17	44	0.0588	39	113	0.1983	267	1322	1.186e+03
	Optimal RL = 17			**Optimal RL = 39**			**Optimal RL = 267**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1322 Hz**		
	Minimum CPU Time = 0.0577 Sec.			**Minimum CPU Time = 0.1982 Sec.**			**Minimum CPU Time = 1.174e+03 Sec.**		
	Average CPU Time = 0.05898 Sec			**Average CPU Time = 0.19858 Sec.**			**Average CPU Time = 1.187e+03 Sec.**		

	LBB–BC								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0581	39	113	0.1987	267	1322	1.165e+03
2	17	44	0.0575	39	113	0.1981	267	1322	1.187e+03
3	17	44	0.0594	39	113	0.213	267	1322	1.185e+03
4	18	42	0.0610	39	113	0.1985	267	1322	1.169e+03
5	17	44	0.0582	39	113	0.1984	267	1322	1.187e+03
6	17	44	0.0574	39	113	0.1845	267	1322	1.184e+03

(*Continued*)

Table 6. (Continued).

	LBB–BC								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
7	17	44	0.0575	41	118	0.1984	267	1322	1.178e+03
8	17	44	0.0581	39	113	0.1982	267	1322	1.190e+03
9	18	42	0.0574	39	113	0.1983	267	1322	1.188e+03
10	17	44	0.0576	39	113	0.1984	267	1322	1.189e+03
	Optimal RL = 17			**Optimal RL = 39**			**Optimal RL = 267**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1322 Hz**		
	Minimum CPU Time = 0.0574 Sec.			**Minimum CPU Time = 0.1845 Sec.**			**Minimum CPU Time = 1.165e+03**		
	Average CPU Time = 0.0582 Sec.			**Average CPU Time = 0.1984 Sec.**			**Average CPU Time = 1.182e+03 Sec.**		

	LBB–BCM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0551	34	117	0.1965	260	1554	1.172e+03
2	17	44	0.0548	39	113	0.1979	260	1554	1.171e+03
3	17	44	0.055	34	117	0.1964	260	1554	1.164e+03
4	17	44	0.0547	34	117	0.1961	260	1554	1.163e+03
5	18	42	0.0552	34	117	0.1847	260	1554	1.153e+03
6	18	42	0.0549	34	117	0.1967	260	1554	1.166e+03
7	17	44	0.0548	34	117	0.1964	260	1554	1.165e+03
8	17	44	0.0547	34	117	0.2110	260	1554	1.161e+03
9	17	44	0.0549	34	117	0.1885	260	1554	1.162e+03
10	17	44	0.0548	34	117	0.1965	260	1554	1.163e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 260**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1554 Hz**		
	Minimum CPU Time = 0.0547 Sec.			**Minimum CPU Time = 0.1847 Sec.**			**Minimum CPU Time = 1.153e+03 Sec.**		
	Average CPU Time = 0.05489 Sec.			**Average CPU Time = 0.1960 Sec.**			**Average CPU Time = 1.164e+03 Sec.**		

Table 7. Performance of proposed FA and MFA algorithms to find Near-OGRs for different channels in a set of 10 trials.

	FA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	18	42	0.5122	34	117	1.0810	260	1554	1.282e+03
2	17	44	0.4378	34	117	1.0200	260	1554	1.267e+03
3	17	44	0.4388	34	117	1.0131	260	1554	1.270e+03
4	17	44	0.4378	34	117	1.0177	260	1554	1.245e+03
5	18	42	0.4379	39	113	1.0179	260	1554	1.280e+03
6	17	44	0.4371	39	113	1.0158	260	1554	1.271e+03
7	18	42	0.3852	34	117	1.0189	260	1554	1.269e+03
8	18	42	0.4369	34	117	1.0144	260	1554	1.268e+03
9	18	42	0.4379	34	117	1.0157	260	1554	1.272e+03
10	17	44	0.4365	39	113	1.0134	260	1554	1.266e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 260**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1554 Hz**		
	Minimum CPU Time = 0.3852 Sec.			**Minimum CPU Time = 1.0131 Sec.**			**Minimum CPU Time = 1.245e+03 Sec.**		
	Average CPU Time = 0.43981 Sec.			**Average CPU Time = 1.02279 Sec.**			**Average CPU Time = 1.269e+03 Sec.**		
	MFA								
	FAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0542	34	117	0.1447	151	1047	1.159e+03
2	17	44	0.0534	34	117	0.1445	151	1047	1.167e+03
3	17	44	0.0489	34	117	0.1442	151	1047	1.166e+03
4	17	44	0.0533	39	113	0.1441	151	1047	1.167e+03

(*Continued*)

Table 7. (Continued).

	MFA								
	FAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
5	18	42	0.0556	39	113	0.1451	151	1047	1.163e+03
6	17	44	0.0567	34	117	0.1442	151	1047	1.164e+03
7	17	44	0.0562	34	117	0.1435	151	1047	1.166e+03
8	17	44	0.0539	34	117	0.1441	151	1047	1.164e+03
9	17	44	0.0529	34	117	0.1432	151	1047	1.165e+03
10	17	44	0.0530	34	117	0.1446	151	1047	1.175e+03
	Optimal *RL* = 17			**Optimal *RL* = 34**			**Optimal *RL* = 151**		
	Optimal *TBW* = 42 Hz			**Optimal *TBW* = 113 Hz**			**Optimal *TBW* = 1047 Hz**		
	Minimum CPU Time = 0.0489 Sec.			**Minimum CPU Time = 0.1432 Sec.**			**Minimum CPU Time = 1.159e+03 Sec.**		
	Average CPU Time = 0.05381 Sec.			**Average CPU Time = 0.14422 Sec.**			**Average CPU Time = 1.166e+03 Sec.**		
	MFA								
	LFA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0519	39	113	0.1446	151	1047	1.168e+03
2	17	44	0.0538	34	117	0.1442	151	1047	1.166e+03
3	18	42	0.0481	34	117	0.1437	151	1047	1.157e+03
4	17	44	0.0549	34	117	0.1436	151	1047	1.176e+03
5	17	44	0.0577	34	117	0.1443	151	1047	1.160e+03

(*Continued*)

Table 7. (Continued).

	MFA								
	LFA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
6	18	42	0.0565	34	117	0.1439	151	1047	1.161e+03
7	17	44	0.0542	39	113	0.1434	151	1047	1.157e+03
8	17	44	0.0519	39	113	0.1435	151	1047	1.169e+03
9	17	44	0.0557	34	117	0.1439	151	1047	1.168e+03
10	18	42	0.0519	34	117	0.1451	151	1047	1.167e+03
	Optimal *RL* = 17			**Optimal *RL* = 34**			**Optimal *RL* = 151**		
	Optimal *TBW* = 42 Hz			**Optimal *TBW* = 113 Hz**			**Optimal *TBW* = 1047 Hz**		
	Minimum CPU Time = 0.0481 Sec.			**Minimum CPU Time = 0.1434 Sec.**			**Minimum CPU Time = 1.157e+03 Sec.**		
	Average CPU Time = 0.05366 Sec.			**Average CPU Time = 0.14402 Sec.**			**Average CPU Time = 1.165e+03 Sec.**		
	MFA								
	LFAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	18	42	0.0511	34	117	0.1387	151	1047	1.088e+03
2	18	42	0.0520	34	117	0.1411	151	1047	1.070e+03
3	18	42	0.0512	34	117	0.1386	151	1047	1.091e+03
4	18	42	0.0529	34	117	0.1388	151	1047	1.095e+03
5	17	44	0.0439	34	117	0.1389	151	1047	1.090e+03
6	17	44	0.0488	34	117	0.1386	151	1047	1.089e+03
7	17	44	0.0519	39	113	0.1385	151	1047	1.092e+03
8	17	44	0.0512	34	117	0.1391	151	1047	1.096e+03
9	17	44	0.0567	34	117	0.1392	151	1047	1.087e+03
10	17	44	0.0518	34	117	0.1385	151	1047	1.091e+03
	Optimal *RL* = 17			**Optimal *RL* = 34**			**Optimal *RL* = 151**		
	Optimal *TBW* = 42 Hz			**Optimal *TBW* = 113 Hz**			**Optimal *TBW* = 1047 Hz**		
	Minimum CPU Time = 0.0439 Sec.			**Minimum CPU Time = 0.1385 Sec.**			**Minimum CPU Time = 1.070e+03 ec.**		
	Average CPU Time = 0.05115 Sec.			**Average CPU Time = 0.1390 Sec.**			**Average CPU Time = 1.089e+03 Sec.**		

Table 8. Performance of proposed BA and MBA algorithms to find near-OGRs for different channels in a set of 10 trials.

	BA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.4349	34	117	1.0168	151	1047	1.289e+03
2	17	44	0.4291	34	117	1.0169	151	1047	1.260e+03
3	17	44	0.4235	39	113	1.0186	151	1047	1.256e+03
4	18	42	0.4248	34	117	1.0216	151	1047	1.258e+03
5	17	44	0.4267	34	117	1.0175	151	1047	1.259e+03
6	18	42	0.4237	34	117	1.0177	151	1047	1.239e+03
7	18	42	0.4101	34	117	1.0158	151	1047	1.242e+03
8	17	44	0.4267	39	113	1.0169	151	1047	1.258e+03
9	17	44	0.4236	39	113	1.0177	151	1047	1.257e+03
10	17	44	0.4221	39	113	1.0178	151	1047	1.261e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.4101 Sec.			**Minimum CPU Time = 1.0158 Sec.**			**Minimum CPU Time = 1.239e+03 Sec.**		
	Average CPU Time = 0.42452 Sec.			**Average CPU Time = 1.01773 Sec.**			**Average CPU Time = 1.258e+03 Sec.**		
	MBA								
	BAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0525	34	117	0.1416	151	1047	1.165e+03
2	17	44	0.0524	34	117	0.1408	151	1047	1.158e+03
3	17	44	0.0522	34	117	0.1412	151	1047	1.157e+03
4	17	44	0.0521	34	117	0.1407	151	1047	1.160e+03
5	17	44	0.0521	34	117	0.1410	151	1047	1.159e+03
6	17	44	0.0519	34	117	0.1411	151	1047	1.155e+03
7	18	42	0.0522	39	113	0.1413	151	1047	1.157e+03
8	17	44	0.0521	34	117	0.1414	151	1047	1.159e+03
9	17	44	0.0520	39	113	0.1408	151	1047	1.158e+03
10	17	44	0.0519	34	117	0.1410	151	1047	1.156e+03

(*Continued*)

Table 8. (Continued).

Trials	MBA								
	BAM								
	n = 6			n = 8			n = 15		
	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0519 Sec.			**Minimum CPU Time = 0.1407 Sec.**			**Minimum CPU Time = 1.155e+03 Sec.**		
	Average CPU Time = 0.05214 Sec.			**Average CPU Time = 0.14109 Sec.**			**Average CPU Time = 1.158e+03 Sec.**		

Trials	MBA								
	LBA								
	n = 6			n = 8			n = 15		
	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0525	39	113	0.1411	151	1047	1.158e+03
2	17	44	0.0522	39	113	0.1389	151	1047	1.159e+03
3	17	44	0.0520	34	117	0.1388	151	1047	1.162e+03
4	17	44	0.0519	34	117	0.1421	151	1047	1.158e+03
5	17	44	0.0518	34	117	0.1367	151	1047	1.154e+03
6	17	44	0.0521	34	117	0.1411	151	1047	1.157e+03
7	17	44	0.0522	34	117	0.1369	151	1047	1.156e+03
8	18	42	0.0520	34	117	0.1367	151	1047	1.165e+03
9	17	44	0.0523	34	117	0.1385	151	1047	1.157e+03
10	18	42	0.0518	34	117	0.1382	151	1047	1.159e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0518 Sec.			**Minimum CPU Time = 0.1367 Sec.**			**Minimum CPU Time = 1.154e+03 Sec.**		
	Average CPU Time = 0.05208 Sec.			**Average CPU Time = 0.1389 Sec.**			**Average CPU Time = 1.159e+03 Sec.**		

Trials	MBA								
	LBAM								
	n = 6			n = 8			n = 15		
	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	18	42	0.0519	34	117	0.1375	151	1047	1.090e+03
2	18	42	0.0511	34	117	0.1376	151	1047	1.089e+03
3	18	42	0.0510	34	117	0.1389	151	1047	1.078e+03
4	17	44	0.0512	34	117	0.1379	151	1047	1.084e+03
5	17	44	0.0513	34	117	0.1374	151	1047	1.085e+03
6	17	44	0.0510	34	117	0.1378	151	1047	1.084 + 03
7	17	44	0.0511	39	113	0.1380	151	1047	1.088e+03
8	17	44	0.0510	34	117	0.1377	151	1047	1.084e+03
9	17	44	0.0500	34	117	0.1379	151	1047	1.086e+03
10	17	44	0.051	39	113	0.1372	151	1047	1.085e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.05 Sec.			**Minimum CPU Time = 0.1372 Sec.**			**Minimum CPU Time = 1.078e+03 Sec.**		
	Average CPU Time = 0.05106 Sec.			**Average CPU Time = 0.13779 Sec.**			**Average CPU Time = 1.085e+03 Sec.**		

Table 9. Performance of Proposed CSA, CSAM, FPA and FPAM Algorithms to find Near-OGRs for different channels in a set of 10 trials.

	CSA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	18	42	0.0517	39	113	0.1378	151	1047	1.150e+03
2	18	42	0.0518	39	113	0.1381	151	1047	1.147e+03
3	17	44	0.0510	39	113	0.1379	151	1047	1.147e+03
4	17	47	0.0490	34	117	0.1378	151	1047	1.148e+03
5	18	42	0.0517	34	117	0.1379	151	1047	1.149e+03
6	17	44	0.0511	34	117	0.1378	151	1047	1.159e+03
7	17	44	0.0515	34	117	0.1375	151	1047	1.146e+03
8	18	42	0.0513	39	113	0.1379	151	1047	1.139e+03
9	17	44	0.0512	34	117	0.1382	151	1047	1.148e+03
10	18	42	0.0570	34	117	0.1381	151	1047	1.146e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0490 Sec.			**Minimum CPU Time = 0.1375 Sec.**			**Minimum CPU Time = 1.139e+03 Sec.**		
	Average CPU Time = 0.05173 Sec.			**Average CPU Time = 0.1379 Sec.**			**Average CPU Time = 1.148e+03 Sec.**		
	CSAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0508	34	117	0.1367	151	1047	1.085e+03
2	18	42	0.0509	34	117	0.1365	151	1047	1.076e+03
3	17	44	0.0511	34	117	0.1362	151	1047	1.077e+03
4	17	44	0.0507	34	117	0.1365	151	1047	1.078e+03
5	17	44	0.0511	34	117	0.1364	151	1047	1.081e+03
6	17	44	0.0490	34	117	0.136	151	1047	1.069e+03

(*Continued*)

Table 9. (Continued).

	CSAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
7	17	44	0.0505	39	113	0.1371	151	1047	1.075e+03
8	17	44	0.0506	34	117	0.1365	151	1047	1.079e+03
9	17	44	0.0502	34	117	0.1364	151	1047	1.078e+03
10	17	44	0.0508	34	117	0.1367	151	1047	1.077e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0490 Sec.			**Minimum CPU Time = 0.1360 Sec.**			**Minimum CPU Time = 1.069e+03 Sec.**		
	Average CPU Time = 0.05057 Sec.			**Average CPU Time = 0.1365 Sec.**			**Average CPU Time = 1.078e+03 Sec.**		

	FPA								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	18	42	0.0505	34	117	0.1367	151	1047	1.135e+03
2	17	44	0.0507	34	117	0.1368	151	1047	1.136e+03
3	17	44	0.0504	34	117	0.1369	151	1047	1.132e+03
4	17	44	0.0498	34	117	0.1371	151	1047	1.135e+03
5	17	44	0.0521	34	117	0.1368	151	1047	1.137e+03
6	17	44	0.0489	34	117	0.1364	151	1047	1.136e+03
7	17	44	0.0504	34	117	0.1369	151	1047	1.137e+03
8	17	44	0.0502	34	117	0.1368	151	1047	1.134e+03
9	17	44	0.0503	39	113	0.1368	151	1047	1.129e+03
10	17	47	0.0508	34	117	0.1369	151	1047	1.133e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0489 Sec.			**Minimum CPU Time = 0.1364 Sec.**			**Minimum CPU Time = 1.129e+03 Sec.**		
	Average CPU Time = 0.05041 Sec.			**Average CPU Time = 0.13681 Sec.**			**Average CPU Time = 1.134e+03 Sec.**		

	FPAM								
	n = 6			n = 8			n = 15		
Trials	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time	RL	TBW (Hz)	CPU Time
1	17	44	0.0499	39	113	0.1349	151	1047	1.032e+03
2	17	44	0.0484	34	117	0.1351	151	1047	1.034e+03
3	17	44	0.0494	34	117	0.1348	151	1047	1.033e+03
4	17	44	0.0491	34	117	0.1353	151	1047	1.030e+03
5	17	44	0.0480	39	113	0.1347	151	1047	1.029e+03
6	17	44	0.0495	34	117	0.1346	151	1047	1.035e+03
7	17	44	0.0492	34	117	0.1341	151	1047	1.029e+03
8	17	44	0.0512	34	117	0.1346	151	1047	1.028e+03
9	17	42	0.0497	34	117	0.1348	151	1047	1.029e+03
10	18	42	0.0493	34	117	0.1349	151	1047	1.026e+03
	Optimal RL = 17			**Optimal RL = 34**			**Optimal RL = 151**		
	Optimal TBW = 42 Hz			**Optimal TBW = 113 Hz**			**Optimal TBW = 1047 Hz**		
	Minimum CPU Time = 0.0480 Sec.			**Minimum CPU Time = 0.1341 Sec.**			**Minimum CPU Time = 1.026e+03 Sec.**		
	Average CPU Time = 0.04937 Sec.			**Average CPU Time = 0.13478 Sec.**			**Average CPU Time = 1.031e+03 Sec.**		

Table 10. Influence of Population Size on the Performance of MBB-BC, FA and MFA to find Near-OGRs for Various Channels.

	MBB–BC																	
	BB–BCM						LBB–BC						LBB–BCM					
	n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14	
Popsize	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)
10	30	73	44	215	229	996	25	77	46	204	221	1166	25	77	44	208	206	1285
20	30	73	44	215	229	996	25	77	46	204	221	1166	25	77	44	208	206	1285
50	30	73	44	215	229	996	25	77	46	204	221	1166	25	77	44	208	206	1285
80	28	74	44	226	229	996	25	77	57	186	221	1166	28	74	55	176	226	993
100	25	81	57	183	229	996	30	73	58	177	221	1166	30	73	44	208	206	1285

	FA					
	n = 7		n = 9		n = 14	
Popsize	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
10	27	73	44	208	206	991
20	27	73	44	208	206	991
50	27	73	44	208	206	991
80	26	77	44	208	206	991
100	25	80	49	206	169	1001

	MFA																	
	FAM						LFA						LFAM					
	n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14	
Popsize	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)
10	25	77	44	206	206	991	27	73	49	206	169	1001	25	77	44	206	127	927
20	25	77	44	206	206	991	26	77	44	206	169	1001	25	77	44	206	127	927
50	25	77	44	206	206	991	26	77	44	206	169	1001	25	77	44	206	127	927
80	27	73	44	206	206	991	25	80	44	206	169	1001	25	77	49	206	127	927
100	28	74	44	206	206	991	25	80	44	206	169	1001	28	74	44	206	127	927

Table 11. Influence of population size on the performance of BA, MBA, CSA, CSAM, FPA and FPAM algorithms to find near-OGRs for various channels.

	BA						MBA																	
							BAM						LBA						LBAM					
	n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14	
Popsizez	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)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
10	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927
20	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927
50	25	77	44	206	127	927	25	77	44	206	127	927	25	77	58	177	127	927	25	77	44	206	127	927
80	25	77	44	206	127	927	27	73	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927
100	28	74	44	206	127	927	27	73	57	183	127	927	27	73	44	206	127	927	25	77	47	185	127	927

	CSA						CSAM						FPA						FPAM					
	n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14		n = 7		n = 9		n = 14	
Popsize	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)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
10	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927
20	25	77	44	206	127	927	25	77	47	185	127	927	25	77	44	206	127	927	25	77	44	206	127	927
50	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927
80	25	77	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927	25	77	55	176	127	927
100	27	73	44	206	127	927	27	73	44	206	127	927	25	77	44	206	127	927	25	77	44	206	127	927

D. *Influence of Increasing Iterations on Total Optical Channel Bandwidth*

The choice of the best maximum iteration for nature-inspired 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. 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 nature-inspired 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-to-optimal values after certain iterations. This is the point where no further improvement is seen. Tables 12-15 illustrate the influence of increasing iterations on the performance of proposed nature-inspired algorithms for various channels. It is noted that the iterations has little effect for low-value marks (such as n = 7 and 8). But for higher-order marks, the iterations has 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, Luong, and Jemai 2010), the maximum numbers of iterations for Tabu search algorithm to find Golomb ruler sequences were set to 10000 and 30000, respectively. The hybrid approach proposed in (Ayari, Luong, and Jemai 2010) to find Golomb ruler sequences the maximum number of iterations set were 100000. In (Bansal 2014), it was noted that to find near-OGRs, GAs and BBO algorithms stabilized in and around 5000 iterations, while hybrid evolutionary algorithms (Dotú and Hentenryck 2005) get stabilized in and around 10000 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-optimal Golomb rulers, stabilized in or around 1000 iterations.

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

Table 16 enlists the ruler length and total occupied channel bandwidth by different sequences obtained from the proposed nature-inspired 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 16. Comparing the experimental results obtained from the proposed algorithms with best-known OGRs and existing algorithms, it is noted that there is a significant

Table 12. Influence of increasing iterations on the performance of MBB-BC algorithms to find near-OGRs for Various Channels.

	TBW (Hz)									
	BB–BCM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	115	212	655	479	689	971	1230	2467	4113	7185
50	74	117	199	346	613	766	1087	2295	3460	6825
100	73	113	186	333	459	581	1048	2176	2745	6660
150	73	113	183	285	388	562	970	1920	2497	6312
250	73	113	183	274	377	549	904	1746	2338	6214
350	73	113	183	274	377	549	736	1477	2143	5817
500	73	113	183	274	377	549	736	1308	2040	5516
600	73	113	183	274	377	549	736	996	2026	4814
700	73	113	183	274	377	549	736	996	1985	3264
800	73	113	183	274	377	549	736	996	1985	2669
900	73	113	183	274	377	549	736	996	1985	2566
1000	73	113	183	274	377	549	736	996	1985	2566
	TBW (Hz)									
	LBB–BC									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	96	174	373	619	671	864	1190	2274	4261	7088
50	74	113	190	583	519	682	970	2138	3623	6767
100	73	113	177	336	490	679	881	1937	3032	6591
150	73	113	177	304	437	567	823	1723	2745	6244
250	73	113	177	258	378	565	768	1583	2592	5819
350	73	113	177	258	378	565	700	1332	2215	5620
500	73	113	177	258	378	565	700	1219	2100	4817
600	73	113	177	258	378	565	700	1166	2061	4019
700	73	113	177	258	378	565	700	1166	1985	3712
800	73	113	177	258	378	565	700	1166	1985	3397
900	73	113	177	258	378	565	700	1166	1985	2872
1000	73	113	177	258	378	565	700	1166	1985	2872

Iterations	TBW (Hz)									
	LBB–BCM									
	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	74	139	199	405	606	758	1134	2333	3525	6887
50	73	113	183	360	461	605	1049	2242	3525	6660
100	73	113	176	295	387	581	876	1861	2901	6187
150	73	113	176	274	378	550	828	1680	2721	5820
250	73	113	176	259	369	520	786	1494	2484	5718
350	73	113	176	259	369	520	725	1290	2233	4779
500	73	113	176	259	369	520	725	1177	2149	4458
600	73	113	176	259	369	520	725	993	1985	4104
700	73	113	176	259	369	520	725	993	1804	3822
800	73	113	176	259	369	520	725	993	1804	3264
900	73	113	176	259	369	520	725	993	1804	2872
1000	73	113	176	259	369	520	725	993	1804	2872

Table 13. Influence of increasing iterations on the performance of FA and MFA to find near-OGRs for various channels.

	TBW (Hz)									
	FA									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	95	149	342	408	682	831	1297	2242	3703	6244
50	73	113	268	336	523	736	1092	2125	2943	6028
100	73	113	206	287	402	711	987	1883	2497	5812
150	73	113	206	249	399	605	935	1680	2338	5626
250	73	113	206	249	391	562	786	1477	2338	5312
350	73	113	206	249	391	562	725	1434	2233	4714
500	73	113	206	249	391	562	725	1288	2215	4112
600	73	113	206	249	391	562	725	1001	2215	3512
700	73	113	206	249	391	562	725	1001	1804	3221
800	73	113	206	249	391	562	725	1001	1804	3100
900	73	113	206	249	391	562	725	1001	1804	2599
1000	73	113	206	249	391	562	725	1001	1804	2599
	TBW (Hz)									
	MFA									
	FAM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	93	124	254	336	664	746	1115	1937	2215	6187
50	73	113	206	297	559	628	1047	1937	2215	5799
100	73	113	206	249	450	558	951	1876	2143	5726
150	73	113	206	249	386	551	823	1477	2143	5594
250	73	113	206	249	386	551	786	1285	2026	5098
350	73	113	206	249	386	551	675	1166	1985	4112
500	73	113	206	249	386	551	675	991	1834	3789
600	73	113	206	249	386	551	675	991	1804	3367
700	73	113	206	249	386	551	675	991	1804	3123
800	73	113	206	249	386	551	675	991	1804	2912
900	73	113	206	249	386	551	675	991	1804	2912
1000	73	113	206	249	386	551	675	991	1804	2912

	TBW (Hz)									
	MFA									
	LFA									
Iterations	n = 18	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16
5	6187	81	121	247	298	671	711	1121	2141	2497
50	5799	73	113	206	280	502	679	1065	1920	2452
100	5726	73	113	206	249	458	581	1048	1883	2143
150	5594	73	113	206	249	378	551	987	1494	2040
250	5098	73	113	206	249	378	551	951	1159	2040
350	4112	73	113	206	249	378	551	673	1114	2100
500	3789	73	113	206	249	378	551	673	1001	1958
600	3367	73	113	206	249	378	551	673	1001	1804
700	3123	73	113	206	249	378	551	673	1001	1804
800	2912	73	113	206	249	378	551	673	1001	1804
900	2912	73	113	206	249	378	551	673	1001	1804
1000	2912	73	113	206	249	378	551	673	1001	1804

	TBW (Hz)									
	MFA									
	LFAM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	206	268	646	611	1230	1723	2149	5775
50	73	113	185	258	489	546	1025	1424	2149	5718
100	73	113	185	249	391	515	915	1219	2061	5509
150	73	113	185	249	378	503	786	1177	1985	5065
250	73	113	185	249	378	503	660	1001	1960	4740
350	73	113	185	249	378	503	660	991	1834	3822
500	73	113	185	249	378	503	660	924	1804	3508
600	73	113	185	249	378	503	660	924	1298	3100
700	73	113	185	249	378	503	660	924	1298	2678
800	73	113	185	249	378	503	660	924	1298	2566
900	73	113	185	249	378	503	660	924	1298	2566
1000	73	113	185	249	378	503	660	924	1298	2566

Table 14. Influence of increasing iterations on the performance of BA and MBA to find Near-OGRs for Various Channels.

	TBW (Hz)									
	BA									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	77	117	342	382	689	870	1156	2187	2519	6111
50	74	113	296	376	593	736	1022	2081	2519	5820
100	74	113	206	249	498	711	923	2081	2484	5775
150	74	113	206	249	386	682	886	1937	2407	5302
250	74	113	206	249	386	503	794	1332	2149	5205
350	74	113	206	249	386	503	660	1332	2149	4809
500	74	113	206	249	386	503	660	1159	1958	3927
600	74	113	206	249	386	503	660	924	1958	3595
700	74	113	206	249	386	503	660	924	1298	3310
800	74	113	206	249	386	503	660	924	1298	3079
900	74	113	206	249	386	503	660	924	1298	3079
1000	74	113	206	249	386	503	660	924	1298	3079

	TBW (Hz)									
	MBA									
	BAM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	206	285	550	670	1120	1583	2347	6024
50	73	113	183	282	450	605	1046	1290	2347	5601
100	73	113	183	249	437	581	935	1246	2215	5258
150	73	113	183	249	386	503	861	1246	2143	5098
250	73	113	183	249	386	503	786	1177	1985	4579
350	73	113	183	249	386	503	660	991	1985	4108
500	73	113	183	249	386	503	660	924	1804	3822
600	73	113	183	249	386	503	660	924	1298	3272
700	73	113	183	249	386	503	660	924	1298	2934
800	73	113	183	249	386	503	660	924	1298	2566
900	73	113	183	249	386	503	660	924	1298	2566
1000	73	113	183	249	386	503	660	924	1298	2566

	TBW (Hz)									
	MBA									
	LBA									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	342	286	509	666	1134	1775	2233	5756
50	73	113	206	283	397	643	970	1390	2233	5723
100	73	113	177	249	395	611	969	1390	2149	5656
150	73	113	177	249	391	503	969	1290	2143	4809
250	73	113	177	249	391	503	880	1193	1960	4488
350	73	113	177	249	391	503	660	1193	1960	4191
500	73	113	177	249	391	503	660	924	1834	3927
600	73	113	177	249	391	503	660	924	1298	3680
700	73	113	177	249	391	503	660	924	1298	3264
800	73	113	177	249	391	503	660	924	1298	2912
900	73	113	177	249	391	503	660	924	1298	2912
1000	73	113	177	249	391	503	660	924	1298	2912

	TBW (Hz)									
	MBA									
	LBAM									
	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	216	267	490	628	1064	1308	2115	5704
50	73	113	185	249	425	612	1064	1166	1960	5704
100	73	113	185	249	386	609	951	1166	1960	5515
150	73	113	185	249	386	503	884	991	1845	4675
250	73	113	185	249	386	503	794	1001	1845	3680
350	73	113	185	249	386	503	660	1001	1540	3488
500	73	113	185	249	386	503	660	924	1365	3213
600	73	113	185	249	386	503	660	924	1298	2912
700	73	113	185	249	386	503	660	924	1298	1894
800	73	113	185	249	386	503	660	924	1298	1894
900	73	113	185	249	386	503	660	924	1298	1894
1000	73	113	185	249	386	503	660	924	1298	1894

Table 15. Influence of increasing iterations on the performance of CSA, CSAM, FPA and FPAM algorithms to find Near-OGRs for various channels.

	TBW (Hz)									
	CSA									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	74	117	342	392	609	681	1047	1544	2061	5601
50	73	113	216	258	499	648	987	1436	2058	5483
100	73	113	206	249	399	606	857	1210	1877	4779
150	73	113	206	249	391	503	756	1210	1647	3822
250	73	113	206	249	391	503	660	987	1427	3272
350	73	113	206	249	391	503	660	924	1323	3221
500	73	113	206	249	391	503	660	924	1298	3083
600	73	113	206	249	391	503	660	924	1298	2806
700	73	113	206	249	391	503	660	924	1298	1894
800	73	113	206	249	391	503	660	924	1298	1894
900	73	113	206	249	391	503	660	924	1298	1894
1000	73	113	206	249	391	503	660	924	1298	1894
	TBW (Hz)									
	CSAM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	183	283	521	693	1048	1439	2059	5515
50	73	113	176	249	489	679	923	1166	1958	5302
100	73	113	176	249	448	503	727	1001	1747	4649
150	73	113	176	249	378	503	669	1001	1544	4458
250	73	113	176	249	378	503	660	996	1497	3673
350	73	113	176	249	378	503	660	924	1365	3488
500	73	113	176	249	378	503	660	924	1298	3079
600	73	113	176	249	378	503	660	924	1298	2725
700	73	113	176	249	378	503	660	924	1298	1894
800	73	113	176	249	378	503	660	924	1298	1894
900	73	113	176	249	378	503	660	924	1298	1894
1000	73	113	176	249	378	503	660	924	1298	1894

	TBW (Hz)									
	FPA									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	262	278	544	653	1025	1286	1985	5483
50	73	113	206	249	392	581	998	1286	1804	5258
100	73	113	206	249	386	503	876	1159	1665	5065
150	73	113	206	249	386	503	774	1012	1542	3927
250	73	113	206	249	386	503	660	956	1355	3698
350	73	113	206	249	386	503	660	924	1323	3412
500	73	113	206	249	386	503	660	924	1298	3100
600	73	113	206	249	386	503	660	924	1298	2678
700	73	113	206	249	386	503	660	924	1298	1894
800	73	113	206	249	386	503	660	924	1298	1894
900	73	113	206	249	386	503	660	924	1298	1894
1000	73	113	206	249	386	503	660	924	1298	1894

	TBW (Hz)									
	FPAM									
Iterations	n = 7	n = 8	n = 9	n = 10	n = 11	n = 12	n = 13	n = 14	n = 16	n = 18
5	73	113	177	292	488	611	1010	1172	1978	5217
50	73	113	176	249	386	609	904	1172	1799	4953
100	73	113	176	249	378	503	786	1001	1667	4740
150	73	113	176	249	378	503	698	993	1539	3822
250	73	113	176	249	378	503	660	991	1342	3665
350	73	113	176	249	378	503	660	924	1321	3405
500	73	113	176	249	378	503	660	924	1298	3062
600	73	113	176	249	378	503	660	924	1298	2566
700	73	113	176	249	378	503	660	924	1298	1894
800	73	113	176	249	378	503	660	924	1298	1894
900	73	113	176	249	378	503	660	924	1298	1894
1000	73	113	176	249	378	503	660	924	1298	1894

Table 16. Performance Comparison of Proposed Nature-Inspired Optimization Algorithms to channel allocation.

			ALGORITHMS									
			Conventional Algorithms				Existing Nature–Inspired Algorithms					
	Best Known OGRs (Bloom and Golomb 1977, Shearer 1990, Rankin 1993, Colannino 2003, Dollas, Rankin, and McCracken 1998, GolombRuler 0000, Shearer 2001, Shearer 0000)		EQC (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		SA (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal, Kumar, and Bhalla 2013)	
n	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	11	6	11	6	11
							7		7		7	
5	11	25	–	–	–	–	12	23	12	23	11	23
		28					13	25	13	24	12	25
								29				
6	17	44	45	140	20	60	17	42	17	42	17	42
		47					18	44	18	43	18	44
		50					21	45	20	44		
		52							21	45		
										49		
7	25	77	–	–	–	–	27	73	27	73	25	73
		81					28	78	29	82	26	74
		87					29	79	30	83	28	77
		90					30	80	31	84	30	81
		95					31	83	32	91		
							32	86	33	95		
								95				

(Continued)

Table 16. (Continued).

	ALGORITHMS													
			Conventional Algorithms				Existing Nature–Inspired Algorithms							
	Best Known OGRs (Bloom and Golomb 1977, Shearer 1990, Rankin 1993, Colannino 2003, Dollas, Rankin, and McCracken 1998, GolombRuler 0000, Shearer 2001, Shearer 0000)		EQC (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		SA (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal, Kumar, and Bhalla 2013)			
n	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	121	34	121	39	113		
							41	126	39	125	41	118		
							42	128	40	127	42	119		
							45	129	42	131				
							46	131						
								133						
9	44	206	–	–	–	–	52	192	49	187	44	179		
							56	193	56	200	45	248		
							59	196	61	201	46	253		
							61	203	62	206	61	262		
							63	225	64	215				
							65							
10	55	249	–	–	–	–	75	283	74	274	77	258		
							76	287						
								301						

(*Continued*)

Table 16. (Continued).

	ALGORITHMS												
			Conventional Algorithms				Existing Nature–Inspired Algorithms						
	Best Known OGRs (Bloom and Golomb 1977, Shearer 1990, Rankin 1993, Colannino 2003, Dollas, Rankin, and McCracken 1998, GolombRuler 0000, Shearer 2001, Shearer 0000)		EQC (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		SA (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal, Kumar, and Bhalla 2013)	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
11	72	386	–	–	–	–	94	395	86	378	72	377
		391					96	456	103	435	105	490
									104	440		456
									114	491		
									118			
12	85	503	231	1441	132	682	123	532	116	556	85	550
							128	581	124	590	91	580
							137	660	138	605		613
13	106	660	–	–	–	–	203	1015	156	768	110	768
							241	1048	171	786	113	753
									187	970		
14	127	924	325	2340	286	1820	206	1172	169	991	221	1166
							228	1177	206	1001		
							230	1285	221	1166		
15	151	1047	–	–	–	–	275	1634	260	1322	267	1322
							298	1653	267	1554		
16	177	1298	–	–	–	–	316	1985	283	1804	316	1985
17	199	1661	–	–	–	–	355	2205	354	2201	369	2201
									369	2208		

(*Continued*)

Table 16. (Continued).

	ALGORITHMS													
			Conventional Algorithms				Existing Nature–Inspired Algorithms					
	Best Known OGRs (Bloom and Golomb 1977, Shearer 1990, Rankin 1993, Colannino 2003, Dollas, Rankin, and McCracken 1998, GolombRuler 0000, Shearer 2001, Shearer 0000)		EQC (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		SA (Kwong and Yang 1997, Randhawa, Sohal, and Kaler 2009)		GAs (Bansal 2014)		BBO (Bansal 2014)		BB–BC (Bansal, Kumar, and Bhalla 2013)	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
18	216	1894	561	5203	493	5100	427	2599	362	2566	427	3079
							463	3079	445	2912		
19	246	2225	–	–	–	–	567	3432	467	3337	584	4101
							597	5067	475	3408		
									584	4101		
20	283	2794	703	7163	703	6460	615	4660	578	4306	691	4941
							673	4826	593	4517		
							680	4905	649	4859		
							691	4941				

	Proposed Nature–Inspired Algorithms															
	BB–BCM		LBB–BC		LBB–BCM		FA		FAM		LFA		LFAM		BA	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
3	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
4	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	23	11	23	11	23	11	23	11	23	11	23	11	23	11	23
	12	25	12	24	12	25	12	24	12	24	12	24	12	24	12	25
			13	25		28	13	25	13				13		13	

(*Continued*)

Table 16. (Continued).

	Proposed Nature–Inspired Algorithms															
	BB–BCM		LBB–BC		LBB–BCM		FA		FAM		LFA		LFAM		BA	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
6	17	42	17	42	17	42	17	42	17	42	17	42	17	42	17	42
	18	44	18	44	18	44	18	44	18	44	18	44	18	44	18	44
		46				46										
7	25	73	25	73	25	73	25	73	25	73	25	73	25	73	25	74
	27	74	30	77	28	74	26	77	27	74	26	77	26	74	28	77
	28	77		81	30	77	27	80	28	77	27	80	27	77		81
	30	79						81				81	28			90
8	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
9	44	183	46	177	44	176	44	206	44	206	44	206	44	185	44	206
	57	215	47	204	55	208	49	208	49		49		47	206		
		226	58	217									49			
				228												
10	55	274	55	258	55	259	55	249	55	249	55	249	55	249	55	249
	58	299	77	321		309										
	74	311		341												
11	72	377	72	378	72	369	72	391	72	386	72	378	72	378	72	386
	105	435	103	439	96	397					103	391	103	386		
						434								391		
12	85	549	85	565	85	520	85	515	85	503	85	503	85	503	85	503
	90	565	90	567	91	551										
					93	550										
13	106	736	109	700	106	725	106	725	106	675	106	673	106	660	106	660
	110	755	111	751	111	744		744	111	725	111	720				
		848		763												
14	229	996	221	1166	206	993	169	991	206	991	169	1001	127	924	127	924
					226	1285	206	1001								
15	267	1322	267	1322	260	1554	260	1554	151	1047	151	1047	151	1047	151	1047
16	316	1985	316	1985	283	1804	283	1804	283	1804	283	1804	177	1298	177	1298

(*Continued*)

Table 16. (Continued).

	Proposed Nature–Inspired Algorithms															
	BB–BCM		LBB–BC		LBB–BCM		FA		FAM		LFA		LFAM		BA	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
17	369	2201	369	2201	355	2205	355	2205	354	2208	354	2208	369	2201	199	1661
18	445	2566	436	2872	436	2872	463	2599	362	2912	445	2566	445	2566	427	3079
19	567	3432	584	4101	467	3337	567	3432	467	3337	475	3408	467	3337	467	3337
20	673	4826	673	4826	649	4517	649	4517	615	4660	615	4660	578	4306	578	4306
															615	4660

	Proposed Nature–Inspired Algorithms													
	BAM		LBA		LBAM		CSA		CSAM		FPA		FPAM	
n	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
4	6	11	6	11	6	11	6	11	6	11	6	11	6	11
	7		7				7				7			
5	11	23	11	23	11	23	11	23	11	23	11	23	11	23
	12	24	12	24	12	24	12	24	12	24	12	24	12	24
	13		13	28	13	25								
6	17	42	17	42	17	42	17	42	17	42	17	42	17	42
	18	44	18	44	18	44	18	44	18	44	18	44	18	44
				47		47		47		47		47		47
				50		50		50						50
7	25	73	25	73	25	73	25	73	25	73	25	73	25	73
	26	77	27	77	26	77	26	77	26	77	26	77	26	77
	27				27		27	80	27		27	80	27	
												81		
8	34	113	34	113	34	113	34	113	34	113	34	113	34	113
	39	117	39	117	39	117	39	117	39	117	39	117	39	117
9	44	183	44	177	44	185	44	206	44	176	44	206	44	176
	57	206	58	206	47	206			47	185			47	185
									55	206			55	206
10	55	249	55	249	55	249	55	249	55	249	55	249	55	249

(Continued)

Table 16. (Continued).

	Proposed Nature–Inspired Algorithms													
	BAM		LBA		LBAM		CSA		CSAM		FPA		FPAM	
n	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)	RL	TBW (Hz)
11	72	386	72	391	72	386	72	391	72	378	72	386	72	378
						391			103	386			103	386
12	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
14	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
16	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
18	445	2566	362	2912	216	1894	216	1894	216	1894	216	1894	216	1894
19	467	3337	475	3408	246	2225	246	2225	246	2225	246	2225	246	2225
20	578	4306	578	4306	283	2794	283	2794	283	2794	283	2794	283	2794
			593	4859										

Table 17. Comparison of average CPU time Taken by proposed optimization algorithms for various channels.

	Algorithms						
	GAs (Bansal 2014)	BBO (Bansal 2014)	BB–BC	BB–BCM	LBB–BC	LBB–BCM	FA
n	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.001	0.000	0.000	0.000	0.000	0.000	0.000
5	0.021	0.020	0.009	0.001	0.001	0.001	0.011
6	0.780	0.7432	0.659	0.0589	0.0584	0.0549	0.4398
7	1.120	1.180	1.170	0.0936	0.0935	0.0919	0.8520
8	1.241	1.239	1.210	0.1986	0.1984	0.1961	1.0227
9	1.711	1.699	1.698	1.3190	1.3170	1.1990	1.4890
10	5.499e+01	5.491e+01	5.450e+01	3.321e+01	3.319e+01	3.160e+01	5.211e+01
11	7.200e+02	7.110e+02	6.990e+02	4.982e+02	4.982e+02	4.782e+02	6.710e+02
12	8.602e+02	8.600e+02	7.981e+02	5.865e+02	5.864e+02	5.645e+02	7.890e+02
13	1.070e+03	1.030e+03	1.020e+03	8.989e+02	8.980e+02	8.563e+02	1.010e+03
14	1.028e+03	1.027e+03	1.021e+03	1.019e+03	1.018e+03	1.000e+03	1.019e+03
15	1.440e+03	1.480e+03	1.291e+03	1.187e+03	1.185e+03	1.163e+03	1.270e+03
16	1.680e+03	1.677e+03	1.450e+03	1.367e+03	1.366e+03	1.287e+03	1.439e+03
17	5.048e+04	5.040e+04	4.075e+04	3.759e+03	3.760e+03	3.542e+03	4.041e+03
18	6.840e+04	6.839e+04	5.897e+04	4.087e+04	4.085e+04	3.998e+04	5.875e+04
19	8.280e+04	8.280e+04	7.158e+04	6.988e+04	6.986e+04	6.442e+04	7.132e+04
20	1.12428e+05	1.1196e+05	1.0012e+05	9.810e+04	9.859e+04	9.356e+04	9.876e+04

	Algorithms										
	FAM	LFA	LFAM	BA	BAM	LBA	LBAM	CSA	CSAM	FPA	FPAM
n	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.001	0.001	0.001	0.011	0.001	0.001	0.001	0.001	0.001	0.001	0.001
6	0.0538	0.0536	0.0512	0.4245	0.0521	0.0522	0.0510	0.0517	0.0505	0.0504	0.0494
7	0.0899	0.0895	0.0870	0.845	0.0869	0.0866	0.0849	0.0819	0.0798	0.0798	0.0748
8	0.1442	0.1440	0.1390	1.0177	0.1410	0.1389	0.1377	0.1379	0.1365	0.1368	0.1348

(*Continued*)

Table 17. (Continued).

	Algorithms										
	FAM	LFA	LFAM	BA	BAM	LBA	LBAM	CSA	CSAM	FPA	FPAM
n	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)	CPU time (Sec.)
9	1.1890	1.1880	1.1770	1.4829	1.187	1.185	1.1681	1.177	1.086	1.1751	1.0781
10	3.151e+01	3.149e+01	3.120e+01	4.290e+01	3.131e+01	3.130e+01	3.109e+01	3.127e+01	3.091e+01	3.121e+01	3.039e+01
11	4.766e+02	4.767e+02	4.656e+02	5.710e+02	4.760e+02	4.740e+02	4.552e+02	4.558e+02	4.521e+02	4.545e+02	4.478e+02
12	5.658e+02	5.652e+02	5.648e+02	6.860e+02	5.651e+02	5.650e+02	5.431e+02	5.342e+02	5.318e+02	5.339e+02	5.227e+02
13	8.750e+02	8.751e+02	8.436e+02	1.000e+03	8.748e+02	8.749e+02	8.435e+02	8.711e+02	8.341e+02	8.631e+02	8.239e+02
14	1.014e+03	1.012e+03	0.981e+03	1.017e+03	1.010e+03	1.010e+03	0.911e+03	1.014e+03	0.890e+03	1.007e+03	0.841e+03
15	1.166e+03	1.165e+03	1.090e+03	1.259e+03	1.157e+03	1.158e+03	1.084e+03	1.149e+03	1.078e+03	1.134e+03	1.031e+03
16	1.342e+03	1.342e+03	1.158e+03	1.429e+03	1.339e+03	1.341e+03	1.154e+03	1.332e+03	1.145e+03	1.230e+03	1.100e+03
17	3.460e+03	3.455e+03	3.320e+03	4.050e+03	3.459e+03	3.457e+03	3.225e+03	3.434e+03	3.211e+03	3.412e+03	3.196e+03
18	4.075e+04	4.076e+04	3.880e+04	5.262e+04	4.072e+04	4.075e+04	3.866e+04	4.067e+04	3.767e+04	4.041e+04	3.645e+04
19	6.687e+04	6.688e+04	6.390e+04	7.118e+04	6.586e+04	6.585e+04	6.288e+04	6.571e+04	5.378e+04	6.542e+04	5.258e+04
20	7.335e+04	7.432e+04	7.110e+04	8.356e+04	7.271e+04	7.270e+04	7.020e+04	7.118e+04	6.974e+04	7.032e+04	6.564e+04

improvement in the ruler length and thus the total occupied channel bandwidth that is, the results gets better.

From Table 16, 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-to-optimal. Second observe that the algorithms BB-BCM and LBB-BC do not find best-known rulers after 7-marks, but finds near-to-optimal rulers for 8 to 20-marks. Algorithm LBB-BCM can find best optimal rulers up to 8-marks, but finds near-to-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-to-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-to-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 nature-inspired algorithms to find near-OGRs, slightly outperforms the algorithms presented in their simplified forms. As illustrated in Table 16 for higher-order marks, the algorithms CSA, FPA, BA and their modified forms outperforms the other proposed and existing algorithms in terms of both the ruler length and total occupied channel bandwidth.

F. *Performance Comparison of Proposed Algorithms in Terms of Computational Time*

Finding Golomb ruler sequences 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 (Distributed.net 0000; Dollas, Rankin, and McCracken 1998; Rankin 1993; Shearer 1998, 2001, 0000). For example, rulers with 20 to 26-marks were found by distributed OGR project (Distributed.net 0000) 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-to-optimal Golomb rulers by the proposed nature-inspired algorithms and their comparison with the computation time taken by existing algorithms (Ayari, Luong, and Jemai 2010; Bansal 2014; Bansal, Kumar, and Bhalla 2013; Distributed.net 0000; Dollas, Rankin, and McCracken 1998; Rankin 1993; Shearer 1990; Soliday, Homaifar, and Lebby 1995). Table 17 reports the average CPU time taken by proposed algorithms to find near-OGRs up to 20-marks. The experimental CPU time taken by BB-BC algorithm to find near-OGRs is not reported in (Bansal, Kumar, and Bhalla 2013). Then, using the same parameter values as mentioned in (Bansal, Kumar, and Bhalla 2013), algorithm BB-BC to find near-OGRs was executed to obtain the average computational CPU time.

In (Soliday, Homaifar, and Lebby 1995), it is identified that to find Golomb ruler sequences 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, 2.07e+04 years for 14-marks, 3.92e+09 years for 16-marks, 1.61e+15 years for 18-marks and 9.36e+20 years for 20-marks ruler. In (Ayari, Luong, and Jemai 2010), it is reported that CPU time taken by Tabu search algorithm to find OGRs is around 0.1 s for 5-marks, 720 s for 10-marks, 960 s for 11-marks, 1913 s for 12-marks and 2516 s (around 41 min) for 13-marks. The OGRs realized by hybrid Genetic algorithm (Ayari, Luong, and Jemai 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 1 h and 100 h, respectively, while 17, 18 and 19-marks OGRs realized in (Rankin 1993) and (Dollas, Rankin, and McCracken 1998), took around 1440, 8600 and 36200 CPU hours (nearly 7 months), respectively, on a Sun Sparc Classic workstation. Also, the near-OGRs realized up to 20-marks by algorithms GA and BBO (Bansal 2014), the maximum execution time was approximately 31 h i.e. nearly 1.3 days, while for BB-BC (Bansal, Kumar, and Bhalla 2013) the maximum execution time was around 28 h i.e. almost 1.1 days.

From Table 17, 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 nature-inspired algorithms, the minimum execution time is reduced to approximately 18 h i.e. less than 1 day. This represents the improvement achieved by the use of proposed optimization algorithms and their modified forms to find near–OGR sequences. From Table 17, it is further observed that algorithm FPAM outperforms the other proposed optimization algorithms in terms of computational time.

Conclusions and Future Work

In this paper, WDM channel allocation algorithm by considering the concept of OGR sequence is presented. Finding either optimal or near-to-optimal

Golomb ruler sequences through conventional computing algorithms is computationally hard problem because as the number of marks increases, the search for OGRs becomes exponentially more difficult. The aim to use nature-inspired algorithms is not necessarily to produce perfect results, but to produce the near-to-optimal results under the given constraints. Even if exact algorithms are able to find optimal or near-optimal rulers, they remain unpractical in terms of computational complexity. This paper presented the application of five recent 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 formulate the nature-inspired algorithms in combination with mutation and Lévy flight strategies. The proposed optimization algorithms have been validated and compared with other existing algorithms to find near-OGRs. It has been observed that modified forms (MBB-BC, MFA, MBA, CSAM and FPAM), 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 ruler length, total optical channel bandwidth and computation time. Simulations and comparison show that the proposed algorithms and their modified forms 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. This implies that FPAM is potentially more superior to all other proposed algorithms in solving such NP-complete problems in terms of both efficiency and success rate.

To date, the research done by (Aggarwal, 2001; Atkinson, Santoro, and Urrutia 1986; Babcock 1953; Bansal 2014; Bansal, Kumar, and Bhalla 2013; Compunity 0000; Forghieri, Tkach, and Chraplyvy 1995; Forghieri et al. 1994; Hwang and Tonguz 1998; Kwong and Yang 1997; Randhawa, Sohal, and Kaler 2009; Saaid 2010; Sardesai 1999; Thing, Shum, and Rao 2004; Tonguz and Hwang 1998) does not show the implementation of their algorithm in real 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.

ORCID

Shonak Bansal http://orcid.org/0000-0002-6551-6011

References

Afshar, M. H., and I. Motaei. 2011. Constrained big bang–big crunch algorithm for optimal solution of large scale reservoir operation problem. *International Journal of Optimization in Civil Engineering* 2:357–75.

Aggarwal, G. P. 2001. *Nonlinear Fiber Optics*. Second ed. San Diego, CA: Academic Press.

http://www.myreaders.info/html/artificial_intelligence.html

Atkinson, M. D., N. Santoro, and J. Urrutia. June 1986. Integer sets with distinct sums and differences and carrier frequency assignments for nonlinear repeaters. *IEEE Transactions on Communications* 34(6):614–17. doi: 10.1109/TCOM.1986.1096587.

Ayari, N., T. V. Luong, and A. Jemai. 2010. A hybrid genetic algorithm for golomb ruler problem. *ACS/IEEE International Conference on Computer Systems and Applications* (AICCSA–2010), Hammamet, Tunisia, pp. 1–4, May 16–19.

Babcock, W. 1953. Intermodulation Interference in Radio Systems. *Bell Systems Technical Journal* 63–73. doi:10.1002/j.1538-7305.1953.tb01422.x.

Bansal, S. 2011 Golomb ruler sequences optimization: Soft computing approaches. *M.Tech. Thesis*, Mullana: Department of Electronics and Communication Engineering, Maharishi Markandeshwar Engineering College, Deemed University,.

Bansal, S. September 2014. Optimal golomb ruler sequence generation for FWM crosstalk elimination: Soft computing versus conventional approaches. *Applied Soft Computing* 22:443–57. doi:10.1016/j.asoc.2014.04.015.

Bansal, S., S. Kumar, and P. Bhalla. 2013. A novel approach to WDM channel allocation: Big Bang–Big crunch optimization. *In the proceeding of Zonal Seminar on Emerging Trends in Embedded System Technologies* (ETECH–2013) *organized by The Institution of Electronics and Telecommunication Engineers* (IETE) (pp. 80–81). Chandigarh: Chandigarh Centre. doi:10.1177/1753193412455776.

Bansal, S., S. Kumar, H. Sharma, and P. Bhalla. May 2011. Golomb ruler sequences optimization: A BBO approach. *International Journal of Computer Science and Information Security (IJCSIS)* Pittsburgh, PA, USA. 9(5):63–71.

Bloom, G. S., and S. W. Golomb. April 1977. Applications of numbered undirected graphs. *Proceedings of the IEEE* 65(4):562–70. doi: 10.1109/PROC.1977.10517.

Blum, E. J., F. Biraud, and J. C. Ribes. 1974. On optimal synthetic linear arrays with applications to radio astronomy. *IEEE Transactions on Antennas and Propagation* 22:108–09. doi:10.1109/TAP.1974.1140732.

Chraplyvy, A. R. October 1990. Limitations on lightwave communications imposed by optical–fiber nonlinearities. *Journal of Lightwave Technology* 8:1548–57. doi: 10.1109/50.59195.

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

http://www.compunity.org/events/pastevents/ewomp2004/jaillet_krajecki_pap_ew04.pdf

Cotta, C., I. Dotu, A. J. Fernandez, and P. V. Hentenryck. September 2007. Local search–based hybrid algorithms for finding golomb rulers. *Kluwer Academic Publishers* Boston. 12 (3):263–91.

Cotta, C., I. Dotú, A. J. Fernández, and P. V. Hentenryck. 9–13 September 2006. A memetic approach to golomb rulers. *Parallel Problem Solving from Nature–PPSN IX* Lecture Notes in Computer Science. 4193:252–61. Springer–Verlag Berlin Heidelberg.

Cotta, C., and A. J. Fernández. 2005. Analyzing fitness landscapes for the optimal golomb ruler problem. In *Evolutionary computation in combinatorial optimization*, ed. J. Gottlieb and G. Raidl, 68–79. vol. 3448, *Lecture Notes in Computer Science* Verlag Berlin: Springer.

Cotta, C., and J. V. Hemert. Recent advances in evolutionary computation for combinatorial optimization. In Cotta, C., and J. V. Hemert (Eds.), *Studies in computational intelligence*, vol. 153. Springer.

Dimitromanolakis, A. June 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., W. T. Rankin, and D. McCracken. January 1998. A new algorithm for golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory* 44(1):379–82. doi: 10.1109/18.651068.

Dotú, I., and P. V. Hentenryck. 2–5 September 2005. A simple hybrid evolutionary algorithm for finding golomb rulers. *Evolutionary Computation, 2005, the 2005 IEEE Congress On* 3:2018–23. doi: 10.1109/CEC.2005.1554943.

Drakakis, K. August 2009. A review of the available construction methods for golomb rulers. *Advances in Mathematics of Communications* 3(3):235–50. doi: 10.3934/amc.

Drakakis, K., and S. Rickard. July 2010. On the construction of nearly optimal golomb rulers by unwrapping costas arrays. *Contemporary Engineering Sciences* 3(7):295–309.

Erol, O. K., and I. Eksin. 2006. A new optimization method: big bang–big crunch. *Advances in Engineering Software* 37,:106–11. doi:10.1016/j.advengsoft.2005.04.005.

Fang, R. J. F., and W. A. Sandrin. 1977. Carrier frequency assignment for non–linear repeaters. *Comsat Technical Review* 7:227–45.

Forghieri, F., R. W. Tkach, and A. R. Chraplyvy. May 1995. WDM systems with unequally spaced channels. *Journal of Lightwave Technology* 13:889–897. doi: 10.1109/50.387806.

Forghieri, F., R. W. Tkach, A. R. Chraplyvy, and D. Marcuse. June 1994. Reduction of four–wave mixing crosstalk in WDM systems using unequally spaced channels. *IEEE Photonics Technology Letters* 6(6):754–56. doi: 10.1109/68.300184.

Galinier, P., B. Jaumard, R. Morales, and G. Pesant. 2001. A constraint–based approach to the golomb ruler problem. *3rd International Workshop on Integration of AI and OR Techniques* (CP–AI–OR 2001), Ashford, Kent, UK.

Gandomi, A. H., X. –. S. Yang, and A. H. Alavi. 2013. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers an International Journal of Simulation–Based Engineering* 29 (1):17–35. Springer Verlang, London. doi:10.1007/s00366-011-0241-y.

Genc, H. M., I. Eksin, and O. K. Erol. 2013. Big Bang–Big crunch optimization algorithm with local directional moves. *Turkish Journal of Electrical Engineering & Computer Sciences* 21:1359–75. doi:10.3906/elk-1106-46.

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization, and machine learning*. USA: Addison Wesley.

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

Hwang, B., and O. K. Tonguz. August 1998. A generalized suboptimum unequally spaced channel allocation technique—Part I: In IM/DDWDMsystems. *IEEE Transactions on Communications* 46:1027–37. doi: 10.1109/26.705403.

Koziel, S., and X. –. S. Yang. 2011. Computational optimization, methods and algorithms. *Studies in Computational Intelligence* 356: 283. Springer.

Kripka, M., and R. M. L. Kripka. 2008. “Big Crunch” optimization method. *Proceedings of the International Conference on Engineering Optimization*, Rio de Janeiro, Brazil: Eng Opt, June 01–05.

Kumbasar, T., E. Yeşil, İ. Eksin, and M. Güzelkaya. 2008. Inverse fuzzy model control with online adaptation via Big Bang–Big crunch optimization. *The 3rd International, Symposium on Communications, Control and Signal Processing* (ISCCSP–2008) (pp. 697–702), Malta, March 12–14.

Kwong, W. C., and G. C. Yang. March 1997. An algebraic approach to the unequal–spaced channel–allocation problem in WDM lightwave systems. *IEEE Transactions on Communications* 45(3):352–59. doi: 10.1109/26.558698.

Lam, A. W., and D. V. Sarwate. 1988. On Optimal Time–Hopping Patterns. *IEEE Transactions on Communications* 36:380–82. doi:10.1109/26.1464.

Lavoie, P., D. Haccoun, and Y. Savaria. 1991. New VLSI architectures for fast soft-decision threshold decoders. *IEEE Transactions on Communications* 39 (2):200–07. doi:10.1109/26.76456.

Leitao, T. June 2004. Evolving the maximum segment length of a golomb ruler. *Genetic and Evolutionary Computation Conference*, USA.

Memarsadegh, N. 11 September 2013. Golomb patterns: introduction, applications, and citizen science game. *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*. New Delhi: Prentice Hall of India Pvt. Ltd.

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

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

Pratap, R. 2010. *Getting started with matlab a quick introduction for scientists and engineers*. New York: Oxford University Press.

Price, K., R. Storn, and J. Lampinen. 2005. *Differential evolution–a practical approach to global optimization*. Berlin, Germany: Springer.

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

Rajasekaran, S., and G. A. Vijayalakshmi Pai. 2004. *Neural networks, fuzzy logic, and genetic algorithms–synthesis and applications*. New Delhi: Prentice Hall of India Pvt. Ltd.

Randhawa, R., J. S. Sohal, and R. S. Kaler. 2009. Optimum algorithm for WDM channel allocation for reducing four–wave mixing effects. *Optik 120* 120:898–904. doi:10.1016/j.ijleo.2008.03.023.

Rankin, W. T. 1993. Optimal golomb rulers: An exhaustive parallel search implementation”, *M.S. Thesis*, Duke University. http://people.ee.duke.edu/~wrankin/golomb/golomb.html

Robinson, J. P. December 1979. Optimum golomb rulers. *IEEE Transactions on Computers* 28 (12):183–84.

Robinson, J. P. May 2000. Genetic search for golomb arrays. *IEEE Transactions on Information Theory* 46(3):1170–73. doi: 10.1109/18.841202.

Robinson, J. P., and A. J. Bernstein. 1967. A class of binary recurrent codes with limited error propagation. *IEEE Transactions on Information Theory* IT–13:106–13. doi:10.1109/TIT.1967.1053951.

Saaid, N. M. May 2010. Nonlinear optical effects suppression methods in WDM systems with EDFAs: A review. International Conference on Computer and Communication Engineering (ICCCE–2010), Kuala Lumpur, Malaysia.

Sardesai, H. P. 23–28 May 1999. A simple channel plan to reduce effects of nonlinearities in dense WDM systems. *Technical Digest. Summaries of papers presented at the Conference on*

Lasers and Electro-Optics. Postconference Edition. CLEO '99. Conference on Lasers and Electro-Optics (IEEE Cat. No.99CH37013), Baltimore, MD, USA, 1999, pp. 183-184, 23–28 May 1999. doi: 10.1109/CLEO.1999.834058.

Shearer, J. B. Smallest known golomb rulers. Mathematics Department, *IBM Research*. http://www.research.ibm.com/people/s/shearer/gropt.html

Shearer, J. B. January 1990. Some new optimum golomb rulers. *IEEE Transactions on Information Theory* 36:183–84. doi: 10.1109/18.50388.

Shearer, J. B. November 1998. Some new disjoint golomb rulers. *IEEE Transactions on Information Theory* 44(7):3151–53. doi: 10.1109/18.737546.

Shearer, J. B. 2001. Golomb ruler table. Mathematics Department, *IBM Research*. http://www.research.ibm.com/people/s/shearer/grtab.html

Singh, K., and S. Bansal. December 2013. Suppression of FWM crosstalk on WDM systems using unequally spaced channel algorithms–A survey. *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)* 3(12):25–31.

Soliday, S. W., A. Homaifar, and G. L. Lebby. 1995. Genetic algorithm approach to the search for golomb rulers", *Proceedings of the Sixth International Conference on Genetic Algorithms* (ICGA–95), Pittsburgh, PA, USA, (pp. 528–35). Morgan Kaufmann.

Storn, R., and K. V. Price. December 1997. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11 (4):341–59. doi: 10.1023/A:1008202821328.

Sugumaran, S., N. Sharma, S. Chitranshi, N. Thakur, and P. Arulmozhivarman. 2013. Effect of four–wave mixing on WDM system and its suppression using optimum algorithms. *International Journal of Engineering and Technology (IJET)* 5 (2):1432–44. Apr–May.

Tabakov, P. Y. 2011. Big Bang–Big crunch optimization method in optimum design of complex composite laminates. *World Academy of Science, Engineering and Technology* 77:835–39.

Thing, V. L. L., M. K. Rao, and P. Shum. October 2003. Fractional optimal golomb ruler based WDM channel allocation. *The 8th Opto-Electronics and Communication Conference (OECC–2003)* 23:631–32.

Thing, V. L. L., P. Shum, and M. K. Rao. December 2004. Bandwidth–Efficient WDM channel allocation for four–wave mixing–effect minimization. *IEEE Transactions on Communications* 52(12):2184–89. doi: 10.1109/TCOMM.2004.838684.

Tonguz, O. K., and B. Hwang. September 1998. A generalized suboptimum unequally spaced channel allocation technique—Part II: In coherent WDM systems. *IEEE Transactions on Communications* 46:1186–93. doi: 10.1109/26.718560.

Yang, X. –. S. 2009. Firefly algorithms for multimodal optimization. *Stochastic Algorithms: Foundations and Applications* (SAGA–2009), Lecture Notes in Computer Science (Vol. 5792, pp. 169–78). Verlag, Berlin : Springer.

Yang, X. –. S. 2010a. *Nature–inspired metaheuristic algorithms*. Second ed. Luniver Press.

Yang, X. –. S. 2010b. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio–Inspired Computation* 2 (2):78–84. doi:10.1504/IJBIC.2010.032124.

Yang, X. –. S., Petridis. 2010c. Firefly algorithm, levy flights and global optimization In *Research and development in intelligent systems XXVI*, ed. M. Bramer and R. Ellis, 209–18. London: Springer.

Yang, X. –. S. April 2010d. A new metaheuristic bat–inspired algorithm. In *Nature inspired coop-erative strategies for optimization (NISCO–2010)*, ed. J. R. Gonzalez et al., vol. 284, 65–74. Springer Berlin: Studies in Computational Intelligence.

Yang, X. –. S. 2011a. Chaos–Enhanced Firefly Algorithm with Automatic Parameter Tuning. *International Journal of Swarm Intelligence Research (IJSIR)* 2 (4):1–11. doi:10.4018/jsir.2011100101.

Yang, X. –. S. 2011b. Review of metaheuristics and generalized evolutionary walk algorithm. *International Journal of Bio–Inspired Computation* 3 (2):77–84. doi:10.1504/IJBIC.2011.039907.

Yang, X. –. S. 2011c. Bat algorithm for multi–objective optimization. *International Journal of Bio–Inspired Computation* 3 (5):267–74. doi:10.1504/IJBIC.2011.042259.

Yang, X. –. S. 2012a. Nature–inspired mateheuristic algorithms: success and new challenges. *Journal of Computer Engineering and Information Technology (JCEIT)* 1 (1):1–3. doi:10.4172/2324-9307.1000e101.

Yang, X. –. S. 2012b. Flower pollination algorithm for global optimization. *Unconventional Computation and Natural Computation 2012*, Lecture Notes in Computer Science (Vol. 7445, pp. 240–49). Berlin : Springer.

Yang, X. –. S. 2013a. Optimization and metaheuristic algorithms in engineering. In *Metaheursitics in water, geotechnical and transport engineering*, ed. X. S. Yang, A. H. Gandomi, S. Talatahari, and A. H. Alavi, 1–23. Elsevier. doi:10.1016/B978-0-12-398296-4.00001-5.

Yang, X. –. S. 2013b. Bat algorithm: literature review and applications. *International Journal of Bio–Inspired Computation* 5 (3):141–49. doi:10.1504/IJBIC.2013.055093.

Yang, X. –. S., and S. Deb. 2009. Cuckoo search via levy flights. *Proc. of World Congress on Nature & Biologically Inspired Computing* (NABIC–2009), USA: IEEE Publications, pp. 210–14.

Yang, X. –. S., and S. Deb. 2010a. Eagle strategy using levy walk and firefly algorithms for stochastic optimization. In *Nature inspired cooperative strategies for optimization (NISCO–2010)*, ed. J. R. Gonzalez, et al., vol. 284, 101–11. Springer Berlin: Studies in Computational Intelligence.

Yang, X. –. S., and S. Deb. 2010b. Engineering optimisation by Cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation* 1 (4):330–43. doi:10.1504/IJMMNO.2010.035430.

Yang, X. –. S., and S. Deb. 2014. Cuckoo search: recent advances and applications. *Neural Computing and Applications* 24 (1):169–74. doi:10.1007/s00521-013-1367-1.

Yang, X. –. S., and A. H. Gandomi. 2012. bat algorithm: a novel approach for global engineering optimization. *Engineering Computations* 29 (5):464–83. doi:10.1108/02644401211235834.

Yang, X. –. S., and X. S. He. 2013. Firefly algorithm: Recent advances and applications. *International Journal of Swarm Intelligence* 1 (1):36–50. doi:10.1504/IJSI.2013.055801.

Yang, X. –. S., M. Karamanoglu, and X. S. He. 2013. Multi–Objective flower algorithm for optimization. *International Conference on Computational Science* (ICCS–2013), Barcelona, Spain, (Vol. 18, pp. 861–68), Procedia Computer Science.

Yang, X. –. S., M. Karamanoglu, and X. S. He. 2014. Flower pollination algorithm: A novel approach for multiobjective optimization. *Engineering Optimization* 46 (9):1222–37. doi:10.1080/0305215X.2013.832237.

Yesil, E., and L. Urbas. 2010. Big Bang–Big crunch learning method for fuzzy cognitive maps. *World Academy of Science, Engineering and Technology* 71:815–24.

Appendix–A

Tables XXIX and XXX illustrates the near–OGR sequences found by proposed nature-inspired optimization algorithms for various marks/channels:

Table A18. Near–OGR sequences found by proposed MBB-BC, FA and MFA algorithms.

	MBB–BC							
	BB–BCM		LBB–BC		LBB–BCM		FA	
n	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
4	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
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
	12	0 1 3 7 12	12	0 1 3 7 12	11	0 2 7 8 11	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
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
	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
	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
	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
	30	0 1 3 7 12 20 30					27	0 1 5 7 15 18 27
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
	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

(*Continued*)

Table A18. (Continued).

	MBB–BC						FA	
	BB–BCM		LBB–BC		LBB–BCM			
n	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	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
	57	0 1 4 6 14 21 32 48 57	47	4 5 11 16 25 33 35 48 51				
			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
	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				
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
	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		
					96	0 2 3 8 17 21 28 50 60 84 96		
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
	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		
13	106	10 12 15 35 47 53 69 80 95 99 108 109 116	109	6 8 9 22 32 40 60 67 79 100 104 109 115	106	2 4 7 27 39 45 61 72 87 91 100 101 108	106	5 12 13 22 26 41 52 68 74 86 106 109 111
	110	3 4 6 26 35 45 50 61 88 95 101 109 113	109	5 7 14 17 37 52 53 77 81 95 103 108 114	111	7 8 11 18 30 44 46 64 73 88 105 113 118	106	2 4 7 27 39 45 61 72 87 91 100 101 108
	110	1 4 11 23 27 48 65 76 78 96 105 110 111	111	2 10 13 15 29 33 50 59 74 84 106 112 113				

(*Continued*)

Table A18. (Continued).

	MBB–BC						FA	
	BB–BCM		LBB–BC		LBB–BCM			
n	Length	Position of Marks	Length	Position of Marks	Length	Position of Marks	Length	Position of Marks
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	15 16 22 24 37 47 75 95 109 139 144 156 185 221	169	0 7 15 24 34 45 57 70 84 99 115 132 150 169
					226	0 1 3 7 12 20 34 44 70 86 109 180 201 226	206	2 3 5 9 17 30 50 67 86 96 126 135 157 208
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
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
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
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
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
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

	MFA					
	FAM		LFA		LFAM	
n	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
4	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
5	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	12	0 1 3 7 12
	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
	18	0 1 3 8 12 18	18	0 1 3 8 12 18	18	0 1 3 8 12 18
7	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 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	26	0 1 7 9 12 22 26	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	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
	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
9	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
	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
					49	1 5 11 12 20 33 36 38 50
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
11	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
			103	1 3 4 12 17 24 34 49 53 77 104	72	0 1 9 19 24 31 52 56 58 69 72
					103	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

(*Continued*)

Table A18. (Continued).

	MFA					
	FAM		LFA		LFAM	
n	Length	Position of Marks	Length	Position of Marks	Length	Position of Marks
13	106	5 12 13 22 26 41 52 68 74 86 106 109 111	106	1 8 9 18 22 37 48 64 70 82 102 105 107	106	0 7 8 17 21 36 47 63 69 81 101 104 106
	111	1 2 4 13 29 34 49 63 71 89 102 106 112	111	3 9 21 22 26 42 52 67 74 76 103 111 114		
14	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
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
16	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	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
18	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	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	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 19. Near-OGR sequences found by proposed BA, MBA, CSA, CSAM, FPA and FPAM algorithms.

	BA		MBA					
			BAM		LBA		LBAM	
n	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
4	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
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
	12	0 1 3 7 12	12	0 1 3 7 12	11	0 2 7 8 11	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
	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 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
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 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 7 13 21 22 25	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	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
	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
9	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	0 3 9 17 19 32 39 43 44
			57	0 1 4 6 14 21 32 48 57	58	0 1 3 8 12 22 28 45 58	47	1 2 4 11 17 22 36 44 48
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
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	0 1 4 13 28 33 47 54 64 70 72
							72	0 1 9 19 24 31 52 56 58 69 72
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

(*Continued*)

Table 19. (Continued).

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

	CSA		CSAM		FPA		FPAM	
n	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
4	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		
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
	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12	12	0 1 3 7 12
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 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	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
7	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	25	0 1 4 10 18 23 25	26	0 1 7 9 12 22 26
	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
	27	0 1 5 7 15 18 27			26	0 1 7 9 12 22 26		
					27	0 1 5 7 15 18 27		
8	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
	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
9	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	0 3 9 17 19 32 39 43 44
			47	1 2 4 11 17 22 36 44 48			47	1 2 4 11 17 22 36 44 48
			55	0 1 3 8 14 18 34 43 55			55	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
11	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 4 13 28 33 47 54 64 70 72	72	0 1 4 13 28 33 47 54 64 70 72
			103	1 3 4 12 17 24 34 49 53 77 104			103	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
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

(*Continued*)

Table 19. (Continued).

	CSA		CSAM		FPA		FPAM	
n	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
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
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
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
18	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	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
20	283	0 24 30 43 55 71 75 89 104 125 127 162 167 189 206 215 272 275 282 283	283	0 24 30 43 55 71 75 89 104 125 127 162 167 189 206 215 272 275 282 283	283	0 24 30 43 55 71 75 89 104 125 127 162 167 189 206 215 272 275 282 283	283	0 24 30 43 55 71 75 89 104 125 127 162 167 189 206 215 272 275 282 283