Novel Approach of Designing Spreading Code using Genetic Algorithm

ARNAB SHASHI HAZARI[1], SIMITA KUNDU[2] AND ABHIJIT CHANDRA[3]
[1]Department of Electrical Engineering, University of Michigan, Ann Arbor, USA
[2]Damodar Valley Corporation, West Bengal, India
[3]Department of Electronics and Telecommunication Engineering, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, India
E-mail: arnabshashi@gmail.com[1], simitakundu@gmail.com[2] and abhijit922@yahoo.co.in[3]

Abstract: - Modern communication systems extensively use Code Division Multiple Access (CDMA) technology that associates one unique code to each user. Therefore the generation of these spreading codes has drawn considerable attention amongst the researchers with a fundamental requirement of having as much distinct members as possible along with a constraint of minimizing the inter-code correlation. Since the operation of trade-off between the number of unique codes and peak cross-correlation value is nothing but a task of optimization; one such intelligent technique called Genetic Algorithm (GA) has been employed in this paper for the design of spreading codes. This has proved the uniqueness of our code generation algorithm over existing ones as it provides flexibility in selecting the number of orthogonal code pairs even when the code-length is constant. Interdependence between the characteristics of the designed code and the parameters of genetic optimization has been revealed by means of a number of design examples. Inherent properties of the GA-optimized spreading code have been analyzed from a number of perspectives. Finally, the worth of the proposed code in current wireless system has been substantiated by comparing its performance with other state-of-the-art spreading codes.

Key-words: - Code Division Multiple Access (CDMA), Genetic Algorithm (GA), Kasami sequence, Orthogonal Gold code, Semi-Orthogonal Modified Large Set Kasami (SOMLK) sequence, Small Set Orthogonal Kasami (SSOK) code, Walsh code

1 Introduction

Modern wireless communication system has been employing Walsh code for spreading the base-band message signal in a downlink synchronous CDMA environment since long because of a number of useful advantages like orthogonality, ease of generation and simplicity of this code family [1]. However, it is heavily challenged by other state-of-the-art orthogonal, near-orthogonal and semi-orthogonal spreading codes [2-4] in various perspectives. Extensive research in this field yields a number of promising contributions [5-9] which outweigh the limitation of Walsh code. Application specific spreading sequences like short length CDMA codes for wireless sensor networks [10] have also been proposed in recent times. Other methods of code design have gradually been proposed in [11-12] to enhance the capacity of the system.

One efficient method for generating orthogonal sets of sequences has been described in [13] which comprises of $(N-1)$ number of orthogonal code sets, each containing N orthogonal sequences of length N. It has been further shown that maximum cross-correlation value improves for sets of size 32 and above as it becomes less than half the sequence length [13].

New sets of Walsh-like nonlinear phase orthogonal codes for synchronous and asynchronous CDMA communication system has been developed in [14]. Authors have claimed that the proposed code outperforms the popularly known Walsh code in an Additive White Gaussian Noise (AWGN) channel and the performances of all the binary codes are comparable to each other in a Rayleigh flat-fading channel.

In this communication, we have made one non-conventional attempt towards the design of CDMA spreading code using an artificially intelligent optimization technique, namely Genetic Algorithm (GA). Design objective of the proposed approach has been the proper trade-off between number of unique sequences and the cross-correlation amongst them. Parameters of the resulting code set have been selected in accordance with the user requirement.

Supremacy of the proposed code has finally been established by referring to a number of existing spreading sequences from the literature.

2 Theoretical Background of Spreading Codes

Various spreading codes have been proposed by a number of researchers across the globe for more than a decade and they are becoming very much acceptable to be utilized in present and future generation CDMA networks. Some of these popularly known codes are briefly described in this section.

2.1 Walsh code

Walsh code set [1] is one of the most popular spreading code families which is presently being used in synchronous downlink Direct Sequence CDMA (DS-CDMA) communication system as all the members in this code set are purely orthogonal to each other. The length of an individual member in Walsh code set is always in the form of 2^n for any even number 'n' and can be recursively constructed as follows:

$$W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix} \text{ With } W_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \text{and}$$

$$W_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad (1)$$

As can be noticed from (1), an N-length Walsh code can serve at most N distinct users simultaneously. System capacity is therefore one area where Walsh code is continuously being challenged by other spreading codes of interest. However, it has already been reported in the literature that Walsh code shows very poor performance in asynchronous multi-user CDMA system [11], [14].

2.2 Orthogonal Gold code

Being urged for enhancing the number of distinct code members, a new type of near-orthogonal code set has been developed from a set of sequences known as Gold code and the new code is called as Orthogonal Gold code [13]. Gold sequences are constructed from a preferred pair of maximal length sequences (m-sequence) by the element-by-element multiplication of one sequence with every time shift of the second sequence. Orthogonal Gold code set enhances the number of distinct code members as described in [13] and eventually produces as high as $N.(N-1)$ number of semi-orthogonal code members each of length $\mathrm{N} = 2^{\mathrm{n}}$.

2.3 Kasami sequence

Kasami sequence demands for significant importance in present generation CDMA system due to its very low cross–correlation values [14] and can be directly generated from m-sequence. There are two different sets for this code, namely Small Set Kasami and Large Set Kasami code. Small Set Kasami sequence is therefore capable of producing $\mathrm{M_{SS}} = 2^{\mathrm{n}/2}$ number of binary sequences each of length $\mathrm{N} = 2^{\mathrm{n}} - 1$ [1-2]. The autocorrelation and cross-correlation values of Small Set Kasami sequence of a particular length N can take only three values, namely $\{-1, -(2^{\mathrm{n}/2}+1), -(2^{\mathrm{n}/2}-1)\}$ [1]. This code has been converted into an even length semi-orthogonal code by using one efficient algorithm [2]. This produces a new type of spreading code named as Small Set Orthogonal Kasami (SSOK) code. Large Set Kasami code has Gold code and Small Set Kasami code as its subsets. It can generate $\mathrm{M_{LS=}}\left(2^{3\mathrm{n}/2} + 2^{\mathrm{n}/2}\right)$ number of unique codes of length $\mathrm{N} = 2^{\mathrm{n}} - 1$ [1], [3]. In an attempt for enhancing the capacity of CDMA system, one semi orthogonal code has been proposed in [3] which consider Large Set Kasami code as the basic sequence to start with. This new code set is termed as Semi-orthogonal Modified Large Set Kasami Sequence (SOMLK).

3. Major Design Challenges

This section briefly highlights the major design challenges faced due to the interdependence of the system parameters in CDMA communication as described in the previous section. Researchers are always in search of a code set that can accommodate a large number of users with minimum possible inter-user interference. Hence it becomes a common practice among the designers to trade off one parameter for another. New code sets are generally formulated based on some deterministic sequences like Pseudo noise sequence, Gold sequence, Small set Kasami sequence, Large set Kasami sequence and consequently the result obtained becomes purely deterministic in the sense that no flexibility exists between the total number of members in the code set and number of orthogonal pairs. Some numerical examples may clarify this point in a better

way. Computation of zero-lag cross-correlation for different 16-length spreading codes shows that Walsh code can generate only 16 codes while all the code members are purely orthogonal (100% orthogonality) to each other [1]. Orthogonal Gold code, on the other hand, can generate 240 codes while all the code members are not purely orthogonal to each other. Out of 57360 possible combinations of cross-correlation, only around 41% of them are zero [13]. This code set exhibits a peak cross-correlation value of 8. SSOK code makes one optimization between the codes generated in [1] and [13]. More specifically, it generates 60 unique codes while the maximum value of cross-correlation drops from that of Orthogonal Gold code. However, orthogonality among code members reduces to a value around 25% only [2]. Further optimization has been carried out during the generation of SOMLK sequence. While producing the total number of unique codes to a value of 95, effectively it increases the peak cross-correlation value among the code members too as compared to others. The biggest advantage that can be identified from its characteristics is that the generated code set shows an orthogonality which is as high as around 49% [3].

In comparison with these previously mentioned spreading codes, proposed code set is more flexible in nature as it is capable of producing variable number of distinct code members with a continuously varying orthogonality for a fixed length of the code. This feature was entirely absent in all of the codes proposed earlier and opens a new door in spreading code design. Detailed analytical comparisons have been illustrated with the help of numerical data and graphical representation in the later section.

4 Genetic Algorithm and Its Application in Designing Spreading Code

4.1 Genetic Algorithm

Biologically inspired algorithms have received considerable attention in many engineering applications of late [15]. These algorithms are influenced by numerous natural phenomenon which may be classified into two categories, namely evolutionary algorithm and swarm optimization. Various such algorithms are available in literature, among which Genetic Algorithm (GA) is the most simplest and easy to implement [11], [16-18].

GA operates through simple cycle of stages like creation of a 'population' of strings, evaluation of each string, selection of best string and the process of genetic manipulation towards the creation of better population of strings [19-20]. These four steps have been summarized by means of a flow chart as in Fig. 1 which illustrates the basic idea of GA implementation.

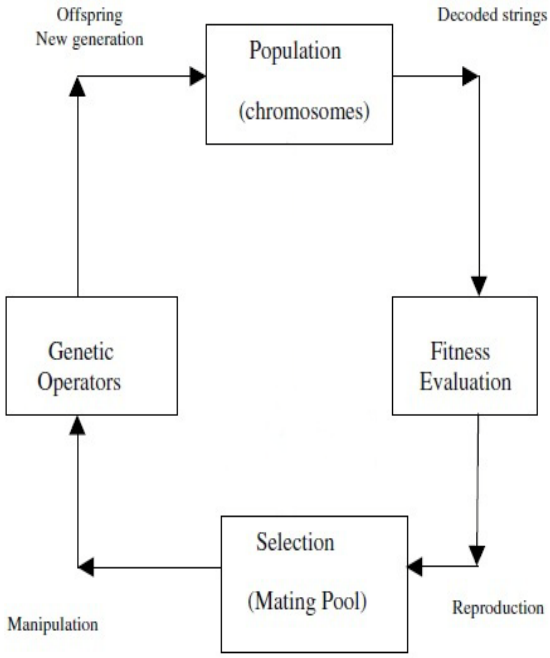

Fig. 1 Cycle of Genetic Algorithm

Each cycle in GA produces a new generation of possible solutions for a given problem. In the first phase, an initial population, describing representatives of the potential solution, is created to initiate the search process. The elements of the population are encoded into bit-strings, called chromosomes. The performance of the strings, often called fitness, is then evaluated with the help of some functions, representing the constraints of the problem under consideration. The analysis of fitness function in context to GA is perhaps best described in [21]. Depending upon the fitness of each chromosome, they are either selected for taking part into subsequent genetic manipulation process or discarded from the population. It should be noted that the selection process is mainly responsible for assuring survival of the best-fit individuals. Detail analysis of selection process has been illustrated lucidly in [22]. Genetic manipulation process consists of two steps, namely cross-over and mutation. In the first step, two selected strings

(chromosomes) exchange their body parts (genes) and the operation is rightly termed as cross-over. In this context, different cross-over mechanisms have been proposed in the literature [23]. The second step in the genetic manipulation process is termed as mutation where the bits (genes) at one or more randomly selected positions of the chromosomes are altered. The mutation process helps the chromosome to overcome trapping at local minima. The offspring produced by the current genetic manipulation process become the next set of population which would be evaluated in similar manner.

4.2 Application of GA in the design of spreading code

Genetic Algorithm (GA) has found widespread applications in the field of machine learning [24], intelligent search [11] and derivative-free optimization problems [25] as reported in the literature. This optimization technique is now being implemented to increase the efficiency of many artificial systems [26]. Moreover, programming environments to apply this technique in such systems are also getting advanced with large diversification [27-28]. It has already been highlighted in the previous section that design of spreading code in CDMA system is purely a task of optimization as a trade-off between total number of distinct codes and the peak cross-correlation value in the code set. Individual code member from the spreading code family may be regarded as a stream of bits which can be encoded into chromosome in a standard GA operation. Purpose of GA in this context is to optimize the set of chromosome (code) in the sense that for a considerably larger size of population, peak cross-correlation between any two arbitrarily chosen codes is within a specified limit. Moreover, the encoding scheme makes it possible to trade one parameter for the other as per requirement. It has been observed that the crossover and mutation operations have significantly altered the cross-correlation value of the offspring from that of the parents. Through the process of selection, the best offspring have been chosen to come up with a more efficient code set. Entire process of GA has been executed iteratively and finally the semi-orthogonal code set for CDMA communication has been produced. Code generation process has been elaborately discussed in the section to follow.

5 Novel Algorithm for the Generation of the Proposed Code

As the proposed algorithm deals with the application of Genetic Algorithm (GA) in the design of CDMA spreading code; initial set of population is one of the most important factors to be considered. It has been reported in many articles that the application of GA in many engineering problems suffers due to the proper choice of initial population which leads to confinement to local minima point. This issue has been seriously taken care by our design process by considering a wide range of such populations. It has been described as follows:
Binary set A consisting of 2^{n} members each of length 2^{m} has been selected initially in a random manner with the constraint $\mathrm{n}-\mathrm{m}\geq 1$. Cross-correlation matrix B of size $(2^{\mathrm{n}}\times 2^{\mathrm{n}})$ is then calculated from the elements of the set A, out of which 2^{n} elements are auto-correlation and the rest are cross-correlations. Every element of B can be represented as:

$$B_{i,j}=A_i * A'_j \tag{2}$$

, where $\mathrm{B_{i,j}}$ is the element in matrix B placed in $\mathrm{i^{th}}$ row and $\mathrm{j^{th}}$ column. A_i signifies all the elements from the $\mathrm{i^{th}}$ row of A where the bit 0 is encoded as -1. The sign '`' and '*' in (2) symbolize transpose and matrix multiplication operation respectively. The element $\mathrm{B_{i,j}}$ therefore is nothing but the cross-correlation between the $\mathrm{i^{th}}$ and the $\mathrm{j^{th}}$ member in the population set computed at zero-lag when $\mathrm{i}\neq\mathrm{j}$ and autocorrelation of the $\mathrm{i^{th}}$ code at zero-lag when $\mathrm{i}=\mathrm{j}$.

Calculation of cross-correlation between all possible members in the code set expedites the selection phase efficiently. In our design process, different thresholds are assigned for best parents and worst parents. These thresholds are identified as $T(b)$ for selecting the best parents and $T(w)$ for selecting the worst parents. Each row of B is then scanned to evaluate the occurrence of zeroes (signifying orthogonal pair) as well as the highest possible cross-correlation (signifying identical pair). If the number of zeroes $(N(z))$ in a row is higher than the given threshold $T(b)$, the code corresponding to that row is selected as one of the best parents and consequently stored in a matrix, called $BEST$.

Conversely, if the number of highest cross-correlation $(N\ (h))$ in any row of B is found to be

greater than the given threshold $T(w)$; the code corresponding to that row is pointed out as one of the worst parents and stored in another separate matrix, called $WORST$.

Formulation of the matrices $BEST$ and $WORST$ enables our algorithm to take part into genetic operations like cross-over and mutation. Although variety of crossover processes is available in literature [29], the simplest one has been employed in our algorithm. This step requires one best parent and one worst parent which are selected from the $BEST$ and $WORST$ matrix randomly. In theory of GA, cross-over operation has been considered as a probabilistic phenomenon in nature with a strong dependence on the crossover probability p_c. However, in this work, we have considered a value of 1 for crossover probability, making the crossover event entirely deterministic. The certainty of cross-over will ensure that the entire system causes significant variations among the parents and offspring. Thus the evolution of the codes is much faster with the incorporation of certainty in the cross-over event. Apart from cross-over probability, the process of cross-over is also influenced by the number of such exchange in the body parts and the position(s) along the length of chromosome where this incident is going to occur. The number of crossover points is chosen in accordance with the following formula:

$$N(c) = 2^{m-2} - 1 \quad (3)$$

The crossover points are mainly found from the following iterative method. The first point is taken near the center of the chromosome. Specifically, if the center point is identified by p^{th}gene, exact position of the crossover point has been chosen randomly with indices ranging from $(p-2)$ to $(p+2)$. Crossover points of these two sub-chromosomes are consequently chosen using the same procedure till the length of the chromosome is less than 6 genes. Once the crossover points are selected, the crossover operation is executed by selecting the body parts from each chromosome, divided by the crossover points. A new chromosome of length 2^m is created by accumulating the segments of two parent chromosomes alternatively. This process creates two new chromosomes of length 2^m which are the inputs to the next step of GA.

Mutation is the next step in genetic evolution which brings variance in the chromosomes resulting in sharp change in the values of cross-correlation. This step, like crossover, is also probabilistic in nature with significant dependence on the mutation probability p_m. Proposed algorithm uses a value of p_m to be 1 as the higher value of mutation probability yields offspring of perceptible variance during the process of evolution. The number of mutation points and its position are important parameters to be selected in this context. For the chromosomes of length 2^m, the number of mutation points $N(m)$ is governed by:

$$N(m) = m - 3 \quad (4)$$

Initially, each chromosome is divided into sub-chromosomes having an equal length of 2^4. Each such sub-chromosome may then be associated with one randomly selected mutation point. Corresponding genes (bits) in the sub-chromosome, identified by their respective mutation points, get inverted through the process of mutation which ensures minimum amount of cross-correlation among the offspring.

One single evolution of GA thus results in two offspring from two parents. These four codes (2 parents and 2 offspring) are subsequently kept in a separate matrix M. Amongst them, best two codes are selected to take part in subsequent genetic operation. Chromosomes (or codes) corresponding to these two numbers are finally returned to the main population from the matrix M. This may cause return of a chromosome which is already present in the population A. The matrix M is again modified during the next iteration.

Since the generation of unique spreading codes is our ultimate objective, distinct code members are then extracted from the modified population matrix yielding a new matrix A^1. All possible cross-correlation values for the elements in matrix A^1 are calculated at zero-lag and consequently stored in matrixB^1. B^1 is constructed in the same way like the computation of B as in equation (2); only difference is that the matrix A is replaced by A^1.

Fitness function of the evolution process has been chosen as the percentage of zeroes in B^1. If this value of the fitness function is greater than a pre-specified threshold, algorithm gets terminated. Otherwise the process of genetic evolution continues iteratively resulting in a modification of A. Flow chart in Fig. 2 illustrates this entire process in brief.

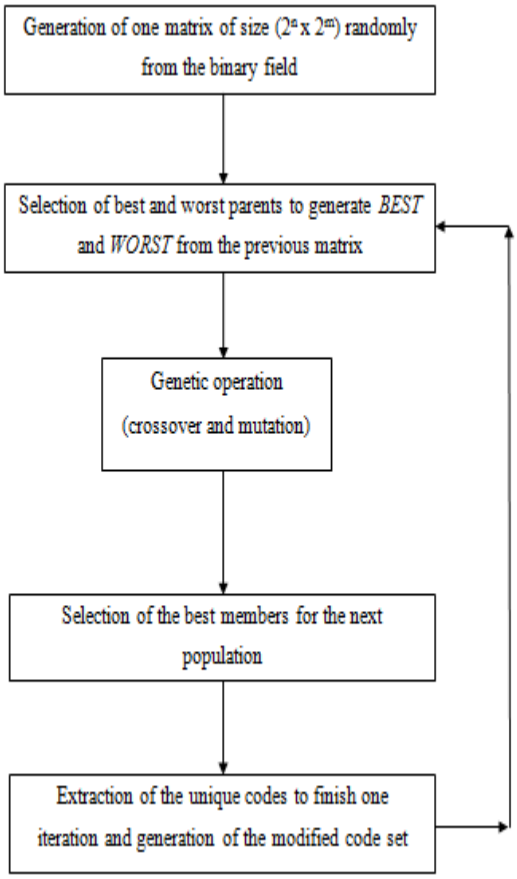

Fig. 2 Flowchart for generation of the proposed code

6 Simulation Result and Analysis

MATLAB 7.10.0.499 software has been used to implement the proposed code generation algorithm which is capable of producing any such code sequences of length 2^m for any positive integer m. Characteristic of our proposed code is quite different from that of the other conventional spreading codes in the sense that it can generate a wide variety of code sets containing different number of code members with a different orthogonality values even when the length of the codes is kept constant. It has also been observed that the proposed code set is strongly dependent on the selection of initial population and allowable maximum number of iterations. These variations have been thoroughly described in the later part of this section. The length of the code has been selected as 2^4 i.e. 16 in our entire analysis. Since GA is a population-based optimization algorithm, selection of basic GA parameters has been explicitly described in Table 1 below.

Table 1. Selection of GA parameters for the proposed design

Name of the parameter	**Choice of the parameter**
Size of population	32/64/128/256/512
Crossover rate	1
Mutation rate	1
Maximum number of iterations	32000

As an initial step to focus on our achievement, we have considered a number of well-recognized spreading sequences for the purpose of comparison with GA-optimized spreading code of length 16. In this context, parameters like number of unique codes generated, peak cross-correlation value, percentage of orthogonal pairs have been taken into consideration. Related characteristics of GA-optimized proposed code have been obtained considering three different initial populations of search space. Number of iterations performed to obtain the given codes by the genetic search algorithm has also been listed accordingly.

Table 2. Comparative analysis among existing & proposed spreading codes

Name of the code	**Number of unique codes**	**Peak cross-correlation value**	**Orthogonal pairs (%)**	**Initial population**	**Number of iteration**
Walsh-Hadamard code [1]	16	0	100	-	-
Orthogonal Gold code [13]	240	8	41.42	-	-
SSOK [2]	60	4	25.34	-	-
SOMLK [3]	95	8	49.36	-	-
Proposed code	242	16	49.55	512	20840

	14	8	95.6	64	32000
	9	0	100	32	17800

Proposed code of Table 2 has been chosen from 128000 code sets available to us. Ideally the programmer can generate infinite such code sets and select the most suitable one for the specific wireless CDMA application. In this article, we have considered three distinct population sizes to initiate the search process. Simulation results show that with an increase in the population sizes, total number of unique codes generated also gets increased with a corresponding decrease in the percentage of orthogonal pairs among them and vice versa. It has even become possible to touch the orthogonality of Walsh code through our genetic approach with an initial population set size of 32; in which case 9 distinct code members are produced. Moreover, starting with a population size of 64, we have come up with a set of codes which can exhibit an orthogonality of around 97% among them. It is noteworthy to mention that except Walsh-Hadamard code, no other spreading code set becomes capable in achieving such mutually orthogonal pairs till date. However, with a population size of 512 due to the inclusion of more number of unique codes, number of orthogonal pair drops to a value of 49.55% which is even higher than what is obtained with Orthogonal Gold [13], SSOK [2], SOMLK [3] code. The set of code vectors obtained with a population size of 512 yields significant advantage in terms of the number of distinct codes produced. Specifically, it provides 93.39%, 0.83%, 75.21%, 60.74% more codes than Walsh [1], Orthogonal Gold [13], SSOK [2] and SOMLK [3] respectively.

Our next two approaches are quite different than the previous one. At first, we achieved a specific value of orthogonality, keeping the number of codes same as Walsh-Hadamard code. Then we have enhanced the value of orthogonality as much as possible by our algorithm at the cost of number of unique codes produced. These two approaches ended with the results shown in Table 3. Since the design objective is to provide a kind of trade off between number of unique codes and the value of orthogonality, the same has been illustrated in Table 3 which lists the optimized binary valued code sequence in decimal form for three different values of orthogonality. Each decimal value in the table represents one 16-bit code. Corresponding decimal values of the proposed code are sorted in descending order in Table 3 for the sake of convenient representation only. Resultant code set can simply be reconstructed from any of the given sets by converting the given decimal values into 16-bit binary vector and putting them row-wise in one matrix.

Table 3. Comparison with Walsh code with typical decimal values

Index	**Walsh-Hadamard code [1]**	**Proposed code**		
		Orthogonality 90%	**Orthogonality 95.6%**	**Orthogonality 100%**
1	65535	63763	63763	64361
2	43690	62926	62926	49605
3	52428	61021	61021	46993
4	39321	54971	54971	40029
5	61680	52660	52520	26460
6	42405	52520	49653	21771
7	50115	49653	45161	14596
8	38550	45161	40275	4983
9	65280	40797	38736	4808
10	43605	38736	32737	-
11	52275	32115	30576	-

12	39270	30576	22744	-
13	61455	22744	21509	-
14	42330	21509	17227	-
15	49980	18331	-	-
16	38505	17227	-	-

Interdependence between the size of the optimized code set and its orthogonal characteristics has been substantiated in Table 3. It has been explicitly pointed out that higher the orthogonality; lesser the number of distinct codes is and vice versa. Moreover, likewise Walsh-Hadamard code, our algorithm becomes successful in producing 16 distinct code members of length 16 with a slight compromise in its orthogonal behavior. Impact of population size on orthogonality and the number of unique codes generated have been reported in Table 4 and Fig. 3 below.

Table 4. Variation of total number of codes with orthogonality

Initial population	**Total number of codes**	**Final orthogonality (%)**	**Iterations performed**
32	10	95.56	32000
64	14	95.6	
128	25	86.33	
256	91	63.47	
512	184	51.4	

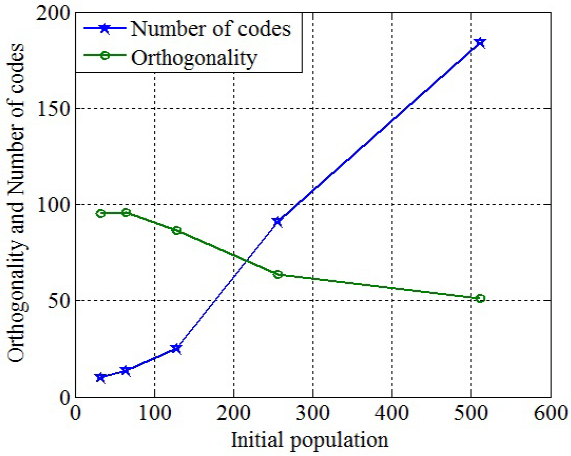

Fig. 3 Variation of orthogonality (%) and number of codes with initial population after the execution of the algorithm

As far as the number of unique codes is concerned, it shows a linear rise with the initial population with a peak value of 184 for a population size of 512. Number of orthogonal pairs, on the other hand, exhibits some different behavior. More specifically, it gets saturated to approximately 50% beyond a population of 256. It will therefore be a clever option to select a higher population size in order to generate more unique codes since orthogonality does not deteriorate too much beyond a certain level.

Following section of our analysis highlights how the orthogonality and total number of distinct codes are modified with the iteration number of the optimization process by means of a number of plots and bar diagrams. The bar diagrams depict the distribution of the cross-correlation values at the end of each process. In this connection, we have provided all such diagrams starting from an initial population of 2^5 and ending with an initial population of 2^9 with an increase in the population by a factor of 2. Those diagrams have been demonstrated in Figure 4 to Fig. 13.

There are various methods to predict the highest value of iterations in any iterative optimization process [7]. In order to encounter all possible situations, the number of iterations has been kept in

the order of thousands. As mentioned previously, the process generates one code set in every iteration which is further optimized in the iterations to come. As a matter of fact, infinite number of such code sets is generated for an infinitely large iteration number.

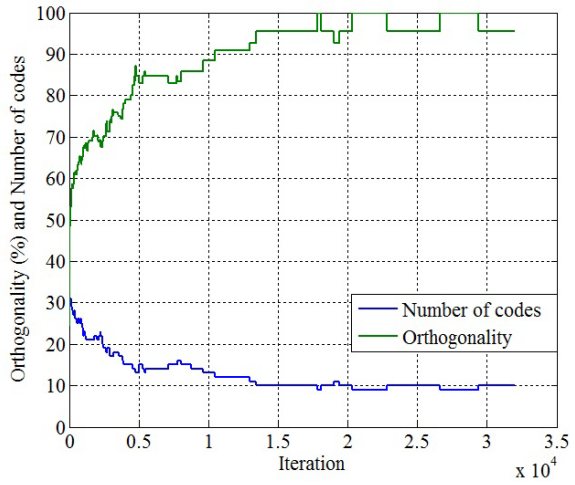

Fig. 4 Variation of number of codes and orthogonality (%) with iteration number for an initial population of 32

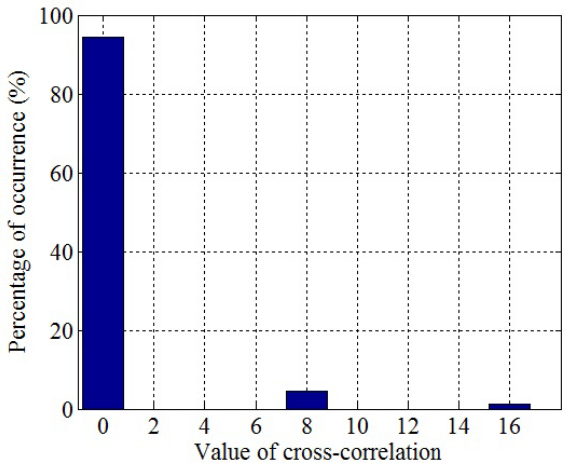

Fig. 5 Histogram of cross-correlation values for an initial population of 32

Looking at Fig. 4 it can be inspected that with an increase in the number of iterations, the value of orthogonality rises sharply up to 5000 iterations beyond which the corresponding rate becomes slow. This rate of increase almost saturates at 14000 iterations. Similar kind of observations may also be made for the number of codes where beyond an iteration number of 14000, reduction in the number of unique codes is almost insignificant. Another important feature may also be extracted from the plot in the sense that we have encountered situations where the value of orthogonality has touched the limit of 100% which, except Walsh code, has never been possible to achieve by any means.

Fig. 5 depicts the distribution of cross-correlations in the entire code set after the termination of the entire process at an iteration of 32000. It can be unambiguously seen that 94.4% of the total cross-correlations are assuming a value of 0 whereas only 1.2% of them are centered on 16. This may be regarded as one of the most promising features of the designed code.

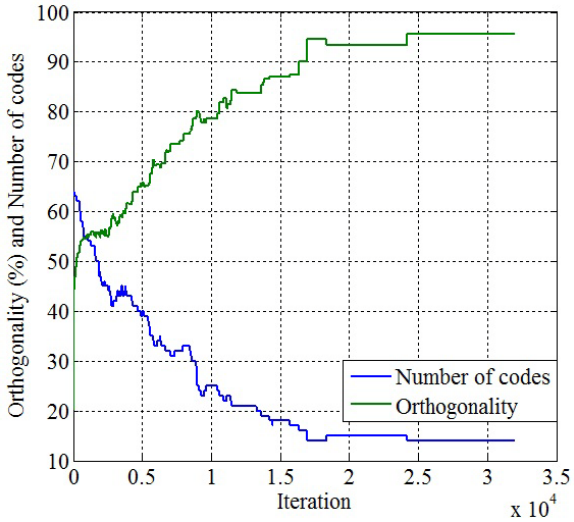

Fig. 6 Variation of number of codes and orthogonality (%) with iteration number for an initial population of 64

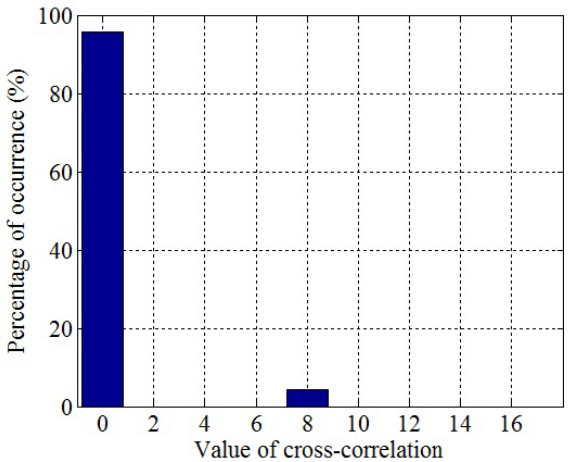

Fig. 7 Histogram of cross-correlation values for an initial population of 64

Variation of number of distinct code members and the orthogonality with the iteration number of GA optimization process for an initial population of 64 shows almost an identical behavior to what is obtained with a population size of 32. Beyond an

iteration of 17000, rate of increase in the orthogonality and decrease in the number of unique codes become insignificant. Since we have started with a higher size of population as compared to Fig. 4, it requires more number of iterations to reach at the point of saturation. Moreover, the maximum possible value of orthogonality can no longer touch the ever hunted limit of 100%; rather confined to value around 96%. On the other hand, a slight enhancement in total number of code members is finally achieved as compared to the design with a population size of 32. As far as the distribution of cross-correlation values is concerned, it can be inferred from Figure 7 that 95.6% of the cross-correlation values are zeros in the final set; while rest are centered at a value of 8. Convergence behavior of the GA-optimized proposed code has further been studied by considering three larger sizes of population set, namely 128, 256 and 512 and demonstrated in Fig. 8 to 13 respectively.

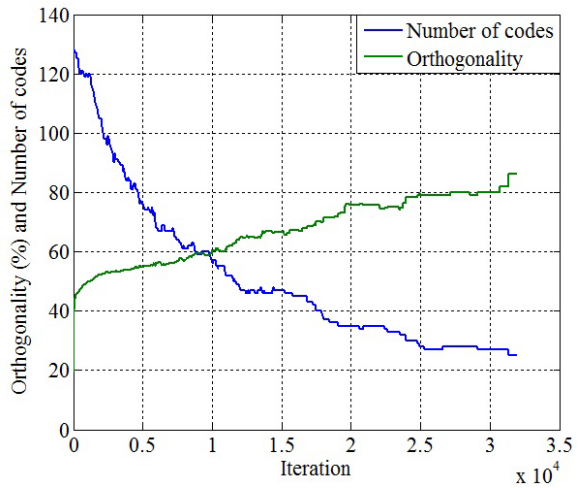

Fig. 8 Variation of number of codes and orthogonality (%) with iteration number for an initial population of 128

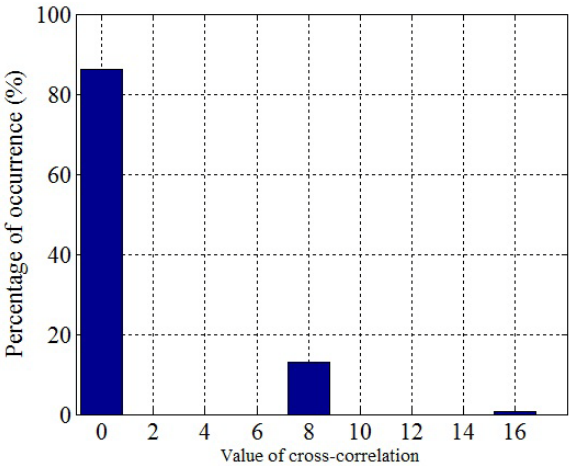

Fig. 9 Histogram of cross-correlation values for an initial population of 128

Characteristic of Fig. 8 is quite different from those of previous figures in the sense that the highest achievable orthogonality is far behind the limit of 100%. Reason behind this behavior is nothing but the incorporation of higher population size in the design process which implicitly is in need for higher iteration numbers for convergence. Nevertheless, finally it results in a code set exhibiting an orthogonality of greater than 80% after the execution of 32000 iterations.

From the bar plot in Fig. 9, it can be clearly seen that the majority (specifically 86.33%) of the cross-correlations are centered on the value of 0 yielding large orthogonal code pairs in the code set. Only 0.7% of them exhibit a value of 16. Therefore the code is highly efficient even when the number of users have started increasing, thus conforming to our statement that the design process retains its supremacy even with a higher number of initial population.

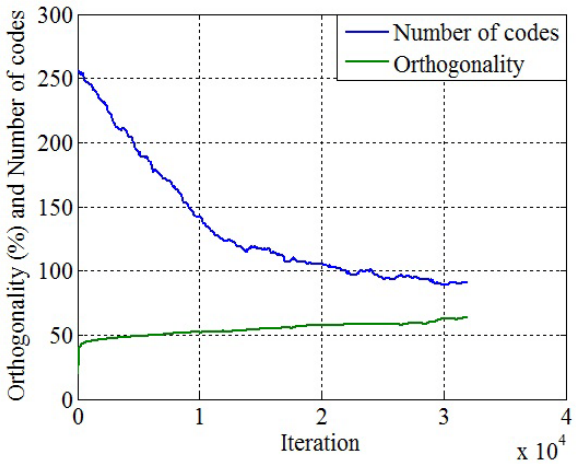

Figure 10: Variation of number of codes and orthogonality (%) with iteration number for an initial population of 256

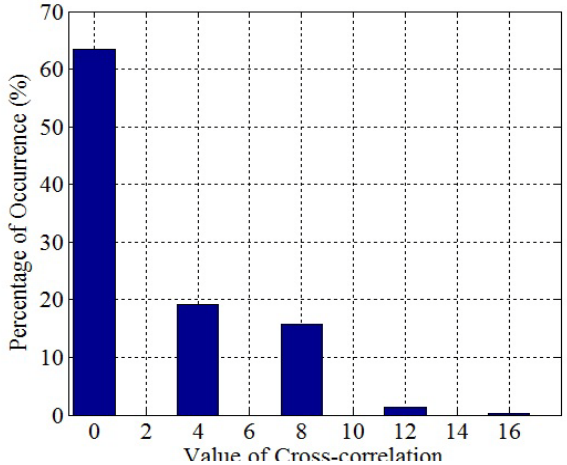

Fig. 11 Histogram of cross-correlation values for an initial population of 256

Generation of GA-optimized proposed spreading code with an initial population of 256 results in a justified trade-off between number of unique codes and achievable orthogonality at the end of specified number of iterations. As can be observed from Fig. 10, it produces a total number of distinct codes very close to 100 with a final orthogonality of greater than 50% after the completion of 32000 iterations. More specifically, this yields 91 unique codes with an associated peak orthogonality of 63.47%. Distribution of cross-correlation values in Fig. 11 suggests that majority of the cross-correlation values are less than or equal to half the sequence length. However, approximately only 1.6% of the total number of cross-correlation pair results in a higher cross-correlation as evident from the histogram plot.

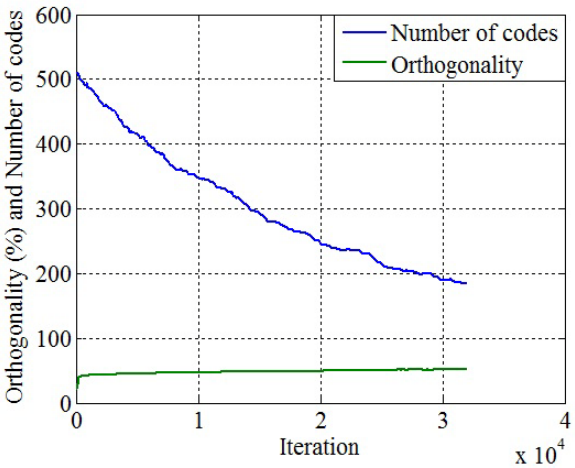

Fig. 12 Variation of number of codes and orthogonality (%) with iteration number for an initial population of 512

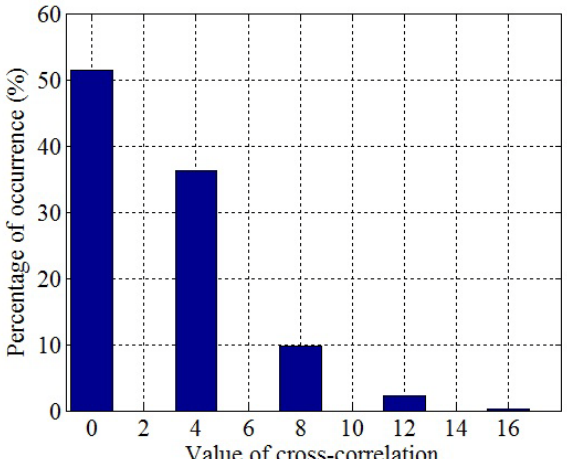

Fig. 13 Histogram of cross-correlation values for an initial population of 512

Fig. 12 reveals one of the most important attributes of our proposed technique. Although the variation of total number of code members and orthogonality with the number of iteration looks very much similar like the one depicted in Fig. 10, initially it produces unique elements as high as 500 in the entire code set with an orthogonality of 40.56%. At an iteration of around 20840, proposed technique results in more than 240 spreading codes of length 16 in the code set with an orthogonality of approximately 50%. This outperforms almost all of the state-of-the-art spreading codes from various perspective of code design. At the end of 32000 iterations, GA-based proposed approach produces 184 distinct codes. Calculation of zero-shift cross-correlation between any two members from this set results in 51.4% orthogonal pair. This substantiates that the quality of the achievable solution gets improved with a corresponding increase in initial set of population and the number of iteration as well.

Frequency of occurrence of the zero-shift cross-correlation among different pairs within the code set reduces with an increase in the magnitude of the correlation as vividly outlined in Figure 13. More specifically, more than 50% of these code pairs produced a cross-correlation value of zero and only 0.21% of them have a value of 16. Cumulative effect of other higher values of cross-correlation like 8 and 12 is within a limit of around 15% and therefore does not affect the overall performance to a greater extent. A relative comparison between the proposed and existing spreading codes in terms of the percentage of occurrence of the magnitude of cross-correlation value has been summarized in Table 5 below.

Table 5. Comparison among various spreading codes in terms of the percentage distribution of cross-correlation

Name of the code	Modulus of cross-correlation					Initial population
	0	4	8	12	16	
Orthogonal Gold code [13]	41.42	46.86	11.72	0	0	-
Modified Walsh-Hadamard code [9]	100	0	0	0	0	-
SSOK [2]	25.34	74.66	0	0	0	-
SOMLK [3]	49.362	36.64	13.998	0	0	-
Proposed code	94.4	0	4.4	0	1.2	32
	95.6	0	4.4	0	0	64
	86.3	0	13	0	0.7	128
	63.5	19.1	15.8	1.29	0.31	256
	51.4	36.3	9.78	2.31	0.21	512

Table 5 summarizes our achievement in brief with a powerful comparison with other state-of-the-art spreading codes. Impact of population size on the attainable performance of the generated code has also been discussed in this regard. As can be seen, except Modified Walsh-Hadamard code [9], our proposed code outperforms other in terms of producing orthogonal pair in the entire code set. Even the minimum orthogonality which has been achieved with an initial population of 512 is significantly higher than what is obtained with [2], [13] and [3]. Moreover, the proposed code with an initial population of 16, 32, 64, 128 and 256 performs better than Orthogonal Gold code [13], SSOK [2] and SOMLK [3] in terms of producing a cross-correlation value of 4. In this context, the performance of our code with a population of 64 is better than the codes described in [13] and [2] and comparable to that of [3] of same length. Comparative analysis of the proposed code with [2], [13] and [3] for other values of cross-correlation can also be substantiated in the same manner.

7 Conclusion

The demand of CDMA communication is on a high rise in the modern era. It has found its widespread applications from high end military communication with very low BER to civilian applications with moderate BER. It has therefore become a challenge for the code designers to develop a new kind of spreading code which may be suitable for a variety of wireless applications. In this communication, we have proposed one novel approach of spreading code design with the aid of GA. It has already been substantiated that the number of unique code members and amount of cross-correlation can be varied in accordance with the proper choice of population size and iteration number of genetic optimization. This flexibility in code design is the unique feature of our algorithm which has not been addressed till date. Simulation results have revealed the capability of our proposed code in achieving two extreme solutions separately corresponding to the highest possible orthogonality among the code members and the largest ever code members for a particular code length. Future research can be progressed by evaluating the performance of the proposed code in a standard CDMA system corrupted by different types of channel noise. Moreover, the overall performance of the generated code may further be enhanced by judiciously tuning the control parameters of GA or even selecting some other optimization techniques.

References:

[1] E. H. Dinan and B. Jabbari, “Spreading codes for direct sequence CDMA and wideband CDMA cellular networks”, IEEE Communication

Magazine, vol. 36, no. 9, pp. 48-54, September, 1998.
[2] A. Chandra and S. Chattopadhyay, "Small Set Orthogonal Kasami Codes for CDMA System", in 4th International Conference on Computers and Devices for Communication, CODEC, December, 2009.
[3] A. S. Hazari, S. Kundu, A. Chandra, "Semi-Orthogonal Modified Large Set Kasami Sequence for CDMA System", In Proc. IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS 2012), pp. 194-197, Bhopal, India, March 1-2, 2012.
[4] T. K. Woo, "Orthogonal code design for quasi-synchronous CDMA", Electronic Letters, vol. 36, no. 19, pp. 1632-1633, September, 2000.
[5] F. Adachi, M. Sawahashi and H. Suda, "Wideband DS-CDMA for next generation mobile communication systems", IEEE Communication Magazine, vol. 36, pp. 56-59, September 1998.
[6] E. A. Bender, "Mathematical Methods in Artificial Intelligence", IEEE Computer Society Press, Los Alamitos, pp. 589-593, 1996.
[7] U. K. Chakraborty and D. G. Dastidar, "Using reliability analysis to estimate the number of generations to convergence in genetic algorithm", Information Processing Letters, vol. 46, pp. 199-209, 1993.
[8] M. Pal, S. Chattopadhyay, "A novel orthogonal minimum cross-correlation spreading code in CDMA system", in Proc. Emerging Trends in Robotics and Communication Technologies (INTERACT), Chennai, 2010.
[9] A. Mitra, "On pseudo-random and orthogonal binary spreading sequence", International Journal of Information and Communication Engineering, vol. 4, no. 6, pp. 447-45, 2008.
[10] R. Poluri and Ali Akansu, "Short Length CDMA Codes for Wireless Sensor Networks", Sarnoff Symposium, IEEE, April 30 – May 2, 2007.
[11] A.N. Akansu, "New binary codes and their comparative BER performance", New Jersey Inst. Technology, Newark, NJ [Online]. Available: http://www.web.njit.edu/~ali/NewCodes.html
[12] P. Kumar, M. Ramesh and S. Chakrabarti, "Performance evaluation of orthogonal/scrambled-orthogonal overloaded DS-CDMA system", IEEE International Conference on Wireless Communication and Networks (WOCN), July, 2007.
[13] H. Donelan and T.O' Farell, "Methods for generating sets of orthogonal sequences", Electronic Letters, vol. 35, pp. 1537-1538, September, 1999.
[14] A.N. Akansu and R. Poluri, "Walsh-like nonlinear phase orthogonal codes for direct sequence CDMA communication", IEEE Transactions on Signal Processing, vol. 55, no. 7, pp. 3800-3806, July, 2007.
[15] D. B. Fogel, "Evolutionary Computation", IEEE Press, Piscataway, NJ, 1995.
[16] J. R. Koza, "Genetic Programming: on the Programming of Computers by Means of Natural Selection", MIT Press, 1992.
[17] M. Mitchell, "An Introduction to Genetic Algorithms", MIT Press, Cambridge, MA, 1996.
[18] T. E. Davis and J. C. Principa, "A Markov chain framework for the simple Genetic Algorithm", Evolutionary Computation, vol. 1, no. 3, pp. 269-288, 1993.
[19] Z. Michalewicz, "Genetic Algorithm + Data Structures = Evolution Programs", 3rd Ed.: Springer-Verlag, Berlin, Germany, 1996.
[20] K. A. De Jong, "An Analysis of Behavior of a Class of Genetic Adaptive Systems", Doctoral dissertation, University of Michigan, 1975.
[21] M. Srinivas and L. M. Patnaik, "Genetic Search: Analysis using fitness moments", IEEE Trans. On Knowledge and Data Engg., vol. 8, no. 1, pp. 120-133, 1996.
[22] U. K. Chakraborty, K. Deb and M. Chakraborty, "Analysis of selection algorithms: a Markov chain approach", Evolutionary Computation, vol. 4, no. 2, pp. 133-167, 1996.
[23] Amit Konar, "Computational Intelligence: Principles, Techniques and Applications", Springer, 2005.
[24] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, MA, 1989.
[25] M. D. Vose and G. E. Liepins, "Punctuated equilibrium in genetic search", Complex Systems, vol. 5, pp. 31-44, 1991.
[26] J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.
[27] J. L. R. Filho and P.C. Trevelan, "Genetic Algorithm Programming Environment", IEEE Computer Society Press, pp. 28-43, June 1994.
[28] J. R. Mc Donell, "Control", In Handbook of Evolutionary Computation, T. Back, D. B. Fogel and Z. Michalewicz (Eds.), IOP and Oxford University Press, New York, 1998.
[29] H. Muehlenbein and U. K. Chakraborty, "Gene pool recombination genetic algorithm and the onemax function", Journal of Computing and Information Technology, vol. 5, no. 3, pp. 167-182, 1997.