A Survey on Crossover Operators

G. PAVAI and T. V. GEETHA, Anna University

Crossover is an important operation in the Genetic Algorithms (GA). Crossover operation is responsible for producing offspring for the next generation so as to explore a much wider area of the solution space. There are many crossover operators designed to cater to different needs of different optimization problems. Despite the many analyses, it is still difficult to decide which crossover to use when. In this article, we have considered the various existing crossover operators based on the application for which they were designed for and the purpose that they were designed for. We have classified the existing crossover operators into two broad categories, namely (1) Crossover operators for representation of applications – where the crossover operators designed to suit the representation aspect of applications are discussed along with how the crossover operators work and (2) Crossover operators for improving GA performance of applications – where crossover operators designed to influence the quality of the solution and speed of GA are discussed. We have also come up with some interesting future directions in the area of designing new crossover operators as a result of our survey.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Control Methods, Problem Solving, and Search—*Heuristic method*

General Terms: Algorithms

Additional Key Words and Phrases: Crossover operator, recombination operator, genetic algorithms, genetic programming, chromosome representation

ACM Reference Format:
G. Pavai and T. V. Geetha. 2016. A survey on crossover operators. ACM Comput. Surv. 49, 4, Article 72 (December 2016), 43 pages.
DOI: http://dx.doi.org/10.1145/3009966

1. INTRODUCTION

Genetic Algorithms are algorithms that mimic the natural evolution process. The underlying principle of two parents producing one or more children by crossover and mutation which resembles human evolution, is common to this algorithm though many variations exist. Some of the terminologies that we should know before we proceed further are as follows:

—*Individual* – A possible solution to the problem considered;
—*Population* – Set of all solutions;
—*Search Space* – Every possible solution that exist to the problem under consideration;
—*Chromosome* – Template or the internal representation details of an individual;
—*Trait (or) Gene* – Features of an individual (e.g., height, color of skin, color of eyes, etc.);
—*Allele* – Settings of trait or possible values for a trait (e.g., height – tall, short, etc.)

This work is supported by the Anna Centenary Research Fellowship.
Authors' address: Department of Computer Science and Engineering, College of Engineering, Guindy, Anna University, Chennai, Tamilnadu, India- 600025; emails: pavai_gops@yahoo.co.in, tv_g@hotmail.com.

DOI: http://dx.doi.org/10.1145/3009966

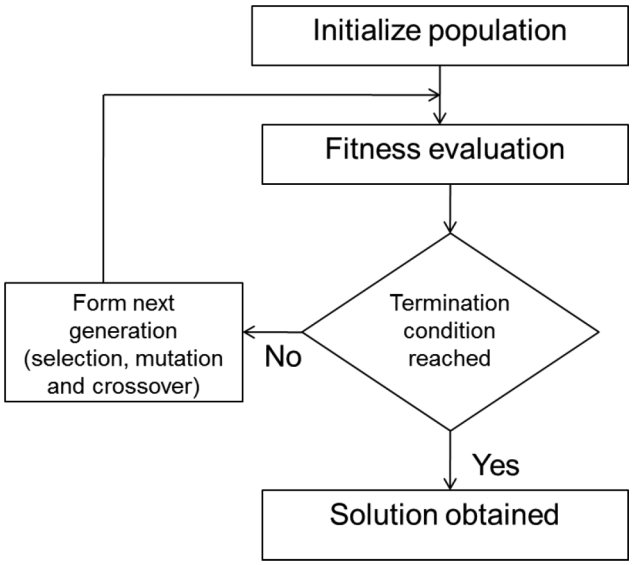

Fig. 1. Genetic Algorithm.

1.1. Basic Components of Genetic Algorithms

Genetic Algorithms have the following basic components as defined by Holland [1992] in his article on the basics of GAs

(1) *An initial set of individuals or initial population*, where the members of the population are representative solutions to the problem for which an optimized solution is required and these solutions are from the search space.
(2) *The representation or the encoding* of the members of the population.
(3) *A fitness function* to calculate the fitness of the individuals in the population so that not all the individuals are considered for producing the next generation and only individuals that are fit enough to survive are considered.
(4) *A selection operation* that selects individuals for mating to produce offspring.
(5) The actual reproduction operation called *the crossover operation* that is responsible for forming offspring where the offspring inherit the characteristics of their parents with the intention of producing better individuals in the succeeding population. This crossover operation is similar to reproduction and biological crossover.
(6) *The mutation operation* that is characterized by modifying one or more genes of an individual follows the crossover operation. Mutation ensures the genetic diversity within the population.
(7) *A termination condition* that suggests when to stop the GA. Some of the commonly used terminating conditions are as follows: A solution satisfying some minimum criteria is found, the specified number of generations is reached, the allocated resources is used, the highest ranked solution's fitness value is unchanged in successive iterations, or any of the combinations of the above. The complete flow of a typical GA is represented in the Figure 1.

In general, any evolutionary algorithm is described by the following equation:

$$p[t + 1] = s(v(p[t]))$$

where,

—p[t] refers to population at generation 't' under representation p[];
—'s' is the selection operation that is responsible for selecting individuals from the mating pool for reproduction;
—'v' is the variation operation which in turn comprises of the crossover (recombination) and the mutation operations.

With a clear description on the gas, the main topic of this survey which is the crossover operator is discussed more in detail in the following sections.

1.2. Importance of Crossover Operators

Ever since the time GAs was invented by Holland [1992], crossover operators are significant as their exploratory force because of their capability to explore solutions over a wider area of the search space. Crossover can be considered identical to the reproduction in natural evolution which keeps the evolution process going. Crossover is the operation that is responsible for creating offspring by which GA is able to explore the search space better and find the globally optimal solution for the problem under consideration. Crossover operation considers two or more individuals of the population as parents to form one or more children by choosing genes from either of the chosen parents or from a combination of both the parents. The crossover operator is said to be the most important operator in the case of GAs because of the building-block hypothesis [Holland 1992]. This hypothesis suggests that certain important building blocks own up the task to produce highly fit individuals in the population. Crossover operation is the most apt operation using which the individuals in the population can mix and match the important building blocks (set of genes) to form better individuals.

The crossover is responsible in sharing information between chromosomes; where the features of two parent chromosomes are combined to form two offspring, aiming to get better chromosomes. Crossover can be considered as a mixer that mixes the solutions, a schemata disrupter that disrupts the schema of the building blocks and an innovator that forms new solutions based on altering or combining the existing building blocks according to Sastry and Goldberg [2002]. The likelihood of the crossover operator being applied to a chromosome depends on the crossover rate. The quality of the solution depends on how effectively the crossover operation is performed. Therefore, crossover operator is considered as one of the GA's "defining characteristics", responsible for improving the performance of GA [Liepins and Vose 1992]. Over the years, crossover operation still remains important for designing solutions to optimization problems. It is this importance of the crossover operator that initially motivated us to carry out this survey. Furthermore, there does not exist any known work that gives information regarding the crossover operators from the application viewpoint, classification based on the representation of the crossover, crossover's role in improving GA performance and the operations of the crossover to help when one needs to choose a crossover from the existing crossovers. Hence, we were motivated to classify the crossover operators from these perspectives.

2. BACKGROUND

The existing surveys on crossover operators mainly discuss the performance of a few crossover operators in relation to a specific problem using a set of evaluation parameters. Some of these surveys are discussed in this section.

Picek et al. [2011] discussed the effects of binary encoded crossover operators in GAs. In this work, the authors aimed to find the best crossover operator for each problem in a

set of optimization problems. The most successful operators for this set of optimization problems are the uniform, single-point, and reduced surrogate crossover whereas, the orthogonal array and hybrid Taguchi crossover operators do not perform as good as the other crossover operators. However, the authors claim that these results are not generic, since there are applications where the crossover operators that performed well in this case behave differently.

There are several surveys that address crossover operators used for a specific application like the Travelling Salesman Problem (TSP) which involves finding the shortest path that covers all the cities exactly only once from a given list of cities and the distance between each other. Misevicius and Kilda [2005] discussed the effects of 12 crossover operators in terms of number of parents, randomization level (explicit mutation, implicit mutation), time complexity, and memory size complexity. They came up with the following results: (1) Longer runs (greater number of generations) are better than the shorter runs (less number of generations) to determine the efficiency of the crossover operators; (2) Crossovers have a high influence on solutions produced by the GA, in spite of using other routines post-recombination, indicating the high potential of the recombination operators; (3) Less disruptive crossover operators are more efficient when compared to their highly disruptive counterparts; (4) Some crossover operators that work well on real-life-like instances gave an opposite performance for random instances and vice-versa; (5) A Partially Mapped Crossover (PMX) crossover operator is highly exceptional as it appears to be superior to the remaining crossover operators in both the random and real-life-like cases.

Ucoluk [2002] compares the performance of two crossover operators namely the PMX and a new inversion permutation crossover, specifically designed for the TSP problem in terms of parameters such as count of cities, count of runs, optimal path length, best path length, average path length, the standard deviation in path length, average iteration count and the standard deviation in iteration count. The comparative study of the results shows that the inversion permutation method outperformed the conventional PMX especially in terms of the convergence rate.

Otman and Jaafar [2011] in their article have discussed the performance of crossover operators specifically designed for the TSP problems. The authors claimed that the optimal solution for BERLIN52 problem (TSP) was given by the OX crossover.

Herrera et al. [2003] have discussed a taxonomy for crossover operators based on the operation creating the offspring but is limited to real coded operators only.

The various surveys discussed above concentrated on either a few selected set of crossover operators or on how each considered crossover operator performed with respect to the evaluation parameters considered for a specific problem. Our survey is entirely different from all the above in that we have proposed a classification for the various existing crossover operators based on all fundamental aspects of crossover operators such as representation of chromosomes used, operators defined, applications considered and the effect of crossover on GA performance such as the quality of the solution and speed. We have also discussed other general crossover-related issues. We proceed with the discussion on the classification in Section 3.

3. THE NEW CLASSIFICATION OF CROSSOVER OPERATORS

The classification that we have derived based on the existing crossover operators is given in Figure 2. Crossover operators are classified based on the representation of the applications for which they were designed and their contribution towards handling GA performance issues such as quality of the solution and speed. The rest of the classification is based on the different representations used and the kind of operation performed in the offspring formation process for various applications for the ‘Crossover operators for representation of applications’ subclass and various techniques adopted to improve the quality of the solution and speed of GA by designing crossover operators

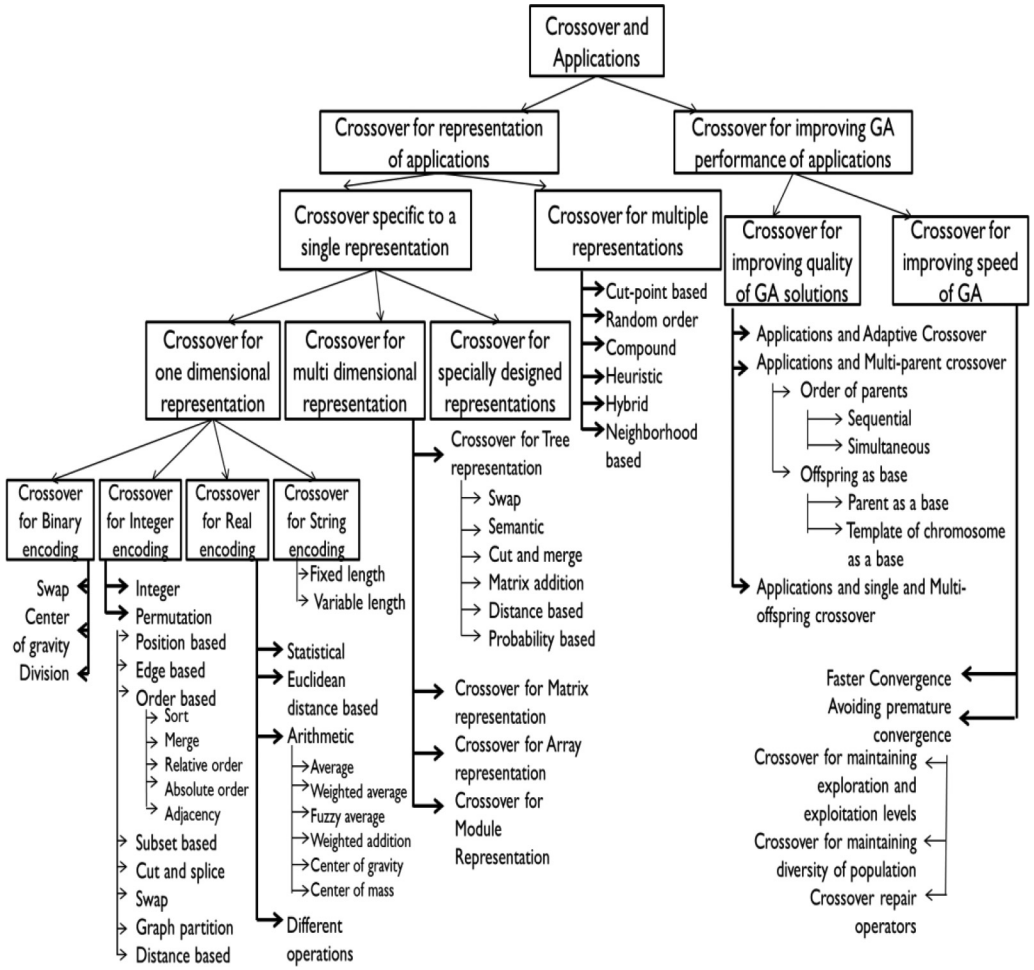

Fig. 2. Classification of crossover operators.

for the 'Crossover for improving GA performance of applications' subclass. The other general issues that form an integral part of any discussion on crossover, including crossover bias, crossover rate, effect of mutation etc., are discussed later. The following sections discuss the proposed taxonomy scheme in detail.

4. CROSSOVER AND APPLICATIONS

We have classified the existing crossover operators based on the representation demanded by GA applications and as contributors to the improvement of quality of the solution and speed of GA. The applications are the optimization problems that can be solved by GAs. Section 4.1 gives the relationship between crossover and representation of applications while Section 4.2 outlines the relationship between crossover and GA performance improvement. Section 4.3 outlines our proposed taxonomy of the existing crossover operators.

4.1. Crossover and Representation for Applications

The application forms the center point of the class 'crossover for representation of applications' in the first level of the classification based on which we have classified some of

the existing crossover operators as shown in Figure 2. Applications and representation go hand-in-hand that whenever GA is chosen as the optimization method for an application, the first question is 'How are the solutions represented?' The representation which suits certain applications demands the design of crossover operators specifically to cater to the chosen representation since not all crossover operators suit all kinds of representation. This is clear from the following sentence "Some forms of crossover operators are more suitable for solving certain problems than others", by Herrera et al. [2005]. Basic crossover operators like the single-point, two-point etc. can be used across applications. However, experiments and results discussed in various articles suggest that application specific crossovers perform better than basic crossovers. Moreover, certain applications are better represented with certain types of encoding. In fact, majority of the crossover operators are designed for the underlying representation which in turn depends on the problem or the application chosen. This was the initial driving force behind the design of new crossover operators. Hence, we have provided a very important place for the design of crossover operators for different representations in our classification of the crossover operators.

4.2. Crossover and GA Performance Improvement

Crossover operators apart from being representation specific, also participate in handling improving different aspects of GA like the quality of the solution and the speed. Performance of GA in terms of the quality of the converged solution needs a special mention since it is a known fact that in cases of very large search spaces, the GA is capable of giving a solution that need not necessarily be optimal. This clearly is an indication of the search space not being searched for the solution efficiently. Crossover is an important operator that helps in searching for the solution efficiently. Hence, modifications to crossover operators were done such that they give optimal solutions for the application considered. Another equally important problem with GAs is their convergence speed. GAs are generally slow to converge to a solution. Jansen and Wegener [2002, 2005] have shown in their work that the choice of the crossover operator reduces the GAs run time and so do Lehre and Yao [2008], Doerr et al. [2008] and Doerr and Theile [2009]. This claim led to the design of many crossover operators that help in reducing the runtime of GAs. However, this improvement in the speed of GAs triggered the problem of premature convergence which essentially reflects the efficiency vs effectiveness tradeoff. This in turn led to the design of new crossover operators that took care of the premature convergence problem as well. This importance of crossover in handling important GA limitations led us to the classification of certain existing crossover operators under the class 'Crossover for improving GA performance of applications' as shown in Figure 2. Now that we have discussed on what led us to the first level of our classification scheme, we describe the detailed description of these two subclasses in the forthcoming sections.

4.3. Outline of the Proposed Taxonomy

This section outlines the flow of the proposed taxonomy that is given in Figure 2, where two important classes of the taxonomy were already outlined in Sections 4.1 and 4.2. The 'Crossover and applications' class forms the root of our taxonomy. It is further subdivided into two classes namely 'Crossover for representation of applications' which describes crossover operators designed to cater to the representation needs of applications (discussed in Section 5) and 'Crossover for improving GA performance of applications' which describes the crossover operators designed for improving GA performance for certain types of applications (discussed in Section 6). We would like to guide GA users to choose the crossover for the application that they want to optimize. If users decide on using a crossover to adapt to a particular representation, the first

branch in the first level is taken. In case they find improving performance of GA is very important for the application in hand (i.e., the application has huge search space, the diversity of the population prevents the GA from converging earlier, there are many local optima apart from the global optima and GA gets easily trapped into one of the local optima, etc.), they can take the second branch of the first level of the classification.

'Crossover for representation of applications' class is further subdivided into crossover designed for single representation (discussed in Section 5.1) and crossover designed for multiple representation, i.e., those crossover operators that can be used across applications (discussed in Section 5.2). Crossover designed for specific representations is further divided based on the dimension of representation into 'Crossover for one dimensional representation' (discussed in Section 5.1.1) and 'Crossover for multi-dimensional representation' (discussed in Section 5.1.2). In addition, crossover operators designed for specially designed representations are classified under the same class (discussed in Section 5.1.3). 'Crossover for one-dimensional representation' discusses the crossover operators designed for various one-dimensional representation schemes like the 'binary encoding, 'integer encoding' (which also describes about crossover operators designed for permutation encoding – a special type of integer encoding), 'real encoding' and 'string encoding' in Sections 5.1.1.1, 5.1.1.2, 5.1.1.3, and 5.1.1.4, respectively. Similarly 'Crossover for multi-dimensional representation' discusses the crossover operators designed for various multi-dimensional representation schemes like the 'tree representation', 'matrix representation', 'array representation,' and 'module representation' in Sections 5.1.2.1, 5.1.2.2, 5.1.2.3 and 5.1.2.4, respectively. The above-mentioned eight sections have further classified the crossover operators falling under them based on the underlying operations that create offspring from the parents. Hence, GA users can identify the type of encoding that better suit the representation of the application considered and chose a crossover operator described under the respective encoding scheme.

The 'Crossover for improving GA performance of applications' section is further subdivided into crossover operators that improve the quality of the GA solution and crossover operators that improve the speed of GA which are discussed in Sections 6.1 and 6.2, respectively. Section 6.1 discusses the role of adaptive crossover (discussed in Section 6.1.1), multi-parent crossover operators (discussed in Section 6.1.2) and single and multi-offspring crossover operators (discussed in Section 6.1.3) in improving the performance of GA. Section 6.1.2 further divides the multi-parent crossover operators based on the operation used for forming offspring from the parents. Section 6.2 is further subdivided into crossover operators for improving speed of GA (discussed in Section 6.2.1) and crossover operators that handle the premature convergence problem – enough speed of convergence such that quality of the solution is not compromised (discussed in Section 6.2.2). Section 6.2.2 is further subdivided into the crossovers using the techniques of maintaining exploration and exploitation levels (discussed in Section 6.2.2.1), maintaining diversity of population (discussed in Section 6.2.2.2) and crossover repair operators (discussed in Section 6.2.2.3) for handling the premature convergence problem. GA users, after identifying the type of improvement required by the application considered, can use one of the crossover operators under one of the methods described in Section 6.

Now we proceed with a detailed discussion on the taxonomy in the subsequent sections.

5. CROSSOVER FOR REPRESENTATION OF APPLICATIONS

Representation is one of the most important part of GA because GAs directly manipulate the chromosomes of the population, which are encoded representations of the problem under consideration and because the representation schema can narrow down

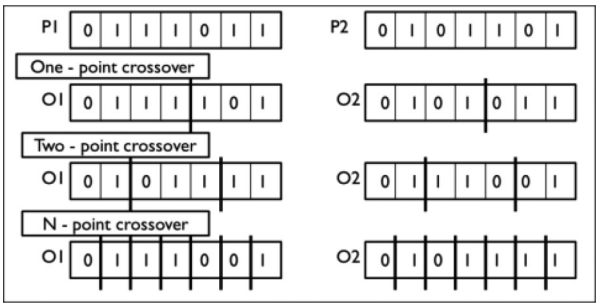

Fig. 3. Basic crossover operators.

the system's observation of its world [Koza 1992]. One of the top levels of our classification is therefore based on representation of the solutions of the problem under consideration. Certain crossover operators are specifically designed for the kind of representation that suits an application and hence are called representation dependent.

Encoding is how a chromosome is represented or is the internal representation of an individual. Different problems demand different types of encoding and there are crossover operators designed specifically for different encoding schemes. Hence, the classification of crossover operators based on representation can be further subdivided into 'crossover for multiple representations' and 'crossover specific to a single representation'.

We start with the description of some of the basic and earliest forms of crossover operators. The description of these basic crossover operators is essential for the understanding of other complex crossover operators. Figure 3 describes some of the basic crossover operators where P1 and P2 are the parents and O1 and O2 are the offspring formed. For the sake of simplicity, the cut-points are shown only on the offspring with a thick vertical line.

One-Point Crossover [Mitchell 1999] – One cut point (also known as crossover point) is considered along the length of the chromosome of the parents such that the Section following the cut point in a parent is swapped with that of the other parent as shown in Figure 3.

Two-Point Crossover [Eshelman et al. 1989; Weise 2009] – Two cut points are considered along the length of the chromosome of each parent such that the next Section is swapped with the other parent as shown in Figure 3.

N-Point Crossover [Sharapov 2007] – N cut points are randomly selected and the alternate sections of the parents are swapped as shown in Figure 3.

Segmented Crossover [Sharapov 2007] – This crossover closely resembles the N-point crossover but for the variation in the number of cut points (number of segments are not constant like the number of cut-points but variable).

Uniform Crossover [Syswerda 1989; Mitchell 1999] – For each position, random decision is made on whether swapping should be done or not.

Shuffle Crossover [Sharapov 2007] – Shuffled parents are formed by the application of a permutation that is selected in a random fashion, followed by N-point crossover on the shuffled parents, which in turn is followed by transformation with inverse permutation (a number is exchanged with another number that is in the position indicated by the first number).

Geometric Crossover [Michalewicz et al. 1996; Moraglio and Poli 2004] – The geometric crossover operator (or the topological crossover operator) is well defined when the solution is defined with distance (geometrically). In this case, once the parents are chosen and the distance measure between the parents is defined, offspring that is

formed within the distance between the two parents are chosen. Some of the existing crossover operations can adapt to this method irrespective of the representation used. The nongeometric crossover [Picek et al. 2011] is similar to geometric crossover. However, the offspring are formed outside the distance between the parents.

Queen Bee Crossover [Karc 2004] – The queen bee crossover operator is where one of the parent chromosomes in the population pool is designated as the queen bee and is made to mate with the rest of the members of the population and the underlying mating procedure follows any one of the basic crossover operators like the one-point or the two-point crossover operators irrespective of the representation.

The above basic crossover operators though designed to suit applications whose individuals can be represented with binary encoding, are still used and compared to the newer operators.

5.1. Crossover Specific to a Single Representation

As we had mentioned earlier, application specific crossover operators are better in performance when compared to the general crossover operators common to more than one representation. Application specific crossover can rather be understood as the crossover designed for the representation of solutions for a particular application and hence are termed as 'crossover specific to a single representation'.

5.1.1. Crossover for One-Dimensional (Linear) Representation. The encoding of the individuals uses some linear order in the case of linear encoding. Linear representation is favored by various applications such as scheduling, TSP etc., Examples of crossover operators that use linear order encoding scheme along with applications that require the encoding are described below.

5.1.1.1. Binary Encoding. Binary representation is where an individual is a string is represented by 0s and 1s. It is simple to represent a chromosome using binary encoding and this representation occupies less space as well. Any application that demands representation of features as either present or absent or can be represented using discrete parameter space can be naturally represented with the binary encoding scheme. According to Herrera et al. [1998], the two main reasons why initial GAs used the binary representation are that binary genes maximizes the implicit parallelism which is one of the important properties of GA and the lower order genes can be represented within a finite population thereby demanding a lower computation cost (cost of calculating fitness for a larger population becomes higher).

The applications that are better represented with the binary encoding are listed below. Any crossover operator that is designed for the binary encoding scheme works well for these applications. The following applications fall under this category:

Classification [Ho et al. 2002] – Considering the classification problem, the solution is better represented as a feature being present or absent in the examples in order to fall under a class. Since it is the present or absent case, binary representation is highly suitable for this class of problems.

Multi-Modal Spin Lattice Problem [Pal 1994] – The spin lattice problem aims to find the minimum energy state of a set of 450 spins with each spin being represented by a four-state spin on two-dimensional lattice, hence making a pair of bits reasonably good (less space consuming) for the representation of each spin.

The binary encoding based crossover operators can be further subclassified into the following types based on the operation used in creating the children.

Swap

The crossover operator swaps the portions of the chromosome that is earlier designated with the help of cut-points (shown with thick vertical lines) to form an offspring (O)

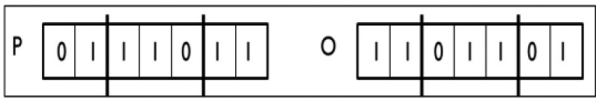

Fig. 4. Binary encoding based crossover operator.

from one parent (P) like in the self-crossover [Pal et al. 1998] as shown in Figure 4 (the sections of the parent before the first cut-point and after the last cut-point are swapped to form the offspring).

Center of Gravity

Supplementary Crossover [Angelov 2001] forms an offspring from two best fit parents in the population by calculating the center of gravity of each gene using the fitness value and the alleles of the parent chromosomes.

Division

Generalized crossover [Coli et al. 1996] works by selecting a random integer and then dividing the decimal equivalents of the binary coded chromosomes to get the remainders which are exchanged and the resultant integer numbers are converted to binary format.

The above crossover operators were designed primarily for applications that were represented easily using binary values. Although various encoding schemes and crossover operators were designed, binary encoding is preferred for its simplicity and lower time consumption and hence crossover operators designed for binary encoding are important.

5.1.1.2. Integer Encoding. Though binary encoding was widely used binary codes are redundant when a variable has a finite number of discrete valid values which is not a power of 2 and hence applications that demanded such values resulted in the design of integer encoding and associated crossover operators. An integer-encoded [Damousis et al. 2004] individual is represented as a string of integers. When there are more discrete variable values that should be represented using binary encoding the length of chromosome becomes higher which in turn reduces the rate of convergence. According to Yuan et al. [2009], this problem can be overcome by using integers for representing the chromosomes instead of binary. There is an improved efficiency with integer representation when compared to the binary representation for applications that can have chromosomes with integer alleles since the need for integer to binary conversion and vice-versa is eliminated by directly representing the alleles in the integer format. The following applications are better represented with integer encoding.

—*Unit Commitment Problem* [Damousis et al. 2004] – "Unit commitment (UC) is an optimization problem used to determine the operation schedule of the generating units at every hour interval with varying loads under different constraints and environments" [Saravanan et al. 2013]. The unit commitment problem can be represented by a set of binary bits to represent the ON/OFF status of operation of the power units over the hours. However, integer encoding is a more straightforward approach, as per Damousis et al. [2004] for this problem.
—*Vehicle Routing with Time Windows* [Nazif and Lee 2010] – The vehicle routing with time windows problem considers vehicles as alleles and paths as genes which obviously fall under the integer category thereby making this representation more suitable for this problem.

Herrera et al. [1998] have identified integer encoding in other situations such as function optimization, parametric design of aircraft, and unsupervised learning of

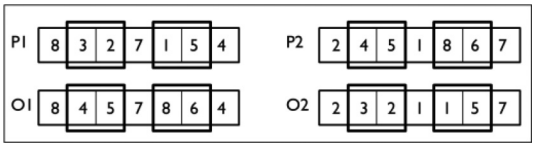

Fig. 5. Integer-encoding-based crossover operator.

neural networks, and the like. Following are some specifically designed integer-coded crossover operators whereas the next section describes the crossover operators designed for permutation encoding. Permutation encoding is a special type of integer encoding where the alleles are integers but a permutation set of integers. Due to this reason, we have classified the permutation-encoding-based crossover operators under integer encoding based crossover operators.

5.1.1.2.1. Integer. Damousis et al. [2004] have designed a multi-point crossover and care is taken to select crossover points that do not cut inside a unit's schedule in the unit commitment problem as shown in the Figure 5. The figure also highlights the units which should not be cut in between. For example, 3 and 2 in P1 of Figure 5 form a unit and hence should not be cut in between according to this crossover operator. However, a unit itself can be moved as a whole while crossing over as shown in the figure.

Optimized crossover [Nazif and Lee 2010] generates as many children as the number of cycles in complete undirected bipartite graph constructed from the parent chromosomes. The child with the optimal fitness value is chosen as the O-Child (Optimum child) which is then used to create the E-Child (Exploratory child) which is used to maintain the diversity of the search space.

Though there are applications that use integer encoding, crossover operators specifically designed for integer encoding are limited since binary-coded crossover operators handle integer encoding in the same way (but for the integer to binary and binary to integer conversion), and hence not many integer-coded crossover operators were developed. The following section discusses permutation-encoding-based crossover operators.

5.1.1.2.2. Permutation Encoding. Many problems can have their solution space encoded as a permutation of set of integers. When traditional crossover operators are applied to permutations, we are likely to get invalid offsprings. Many problems can have their solution space encoded as a permutation of set of integers. The permutation encoding differs from the integer encoding because the sequence of genes is important in the case of permutation encoding whereas, integer encoding does not care about the actual sequence of genes. Permutation encoding is used when the parameter space is discrete and the sequence is important for the application [Fox and McMahon 1991].

The following applications are better represented with the Permutation encoding:

—*TSP* [Eiben et al. 1994; Moraglio 2007; Otman and Jaafar 2011; Agarwal and Singh 2013; Kumar et al. 2013] – Given a list of cities and distance between them, the Travelling Salesman Problem (TSP) finds the shortest route that covers all the cities exactly once and returns to the starting city. Considering the TSP problem, the cities are numbered with integers and certain sequence of cities is important for determining the solution thereby making such kind of problems more suitable for the permutation representation.
—*Graph Coloring Problem* [Eiben et al. 1994; Mumford 2006] – This problem is solved by finding a minimum set of colors for all the vertices in a graph such that the adjacent vertices are not having the same color. The number of vertices forms the number of genes and the colors numbered as integers form the alleles.

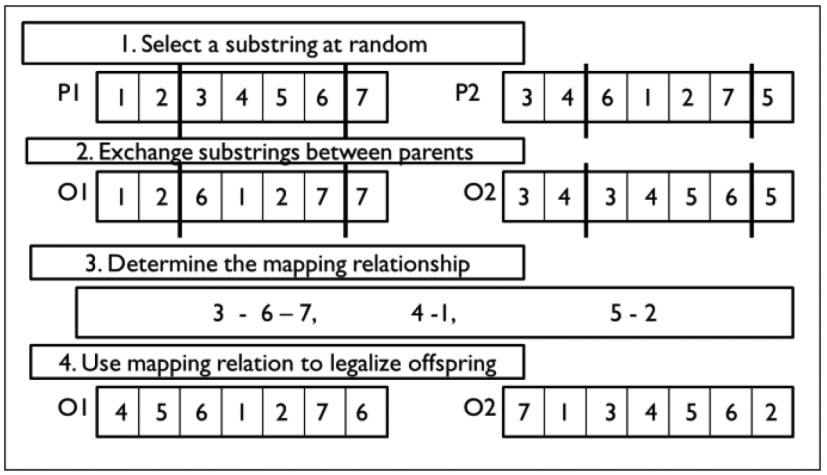

Fig. 6. Permutation encoding based crossover operator.

—*Quadratic Assignment Problem* [Misevicius and Kilda 2005] – This problem deals with assigning the n facilities to n locations in order to minimize the sum of the distances between locations (similar to TSP) such that the cost function is a quadratic inequality.

—*N-Queens Problem* [Moraglio 2007] – This problem is about finding N locations for queens of an $N \times N$ chessboard without threatening each other. The N columns form the genes of the chromosome whereas the alleles for each gene are the N row positions for each column.

Permutation-based crossover operators range from general-purpose operators working on a wide variety of sequencing problems, like the Partially Matched Crossover [Goldberg and Lingle 1985], to specialized operators working on specific problems, for example the sequencing problem subgroup [Fox and McMahon 1990] like TSP, where Edge Recombination Crossover was designed for TSP [Whitley et al. 1991]. Crossover operators using permutation encoding are further subclassified into the following classes based on the underlying operation creating the offspring.

5.1.1.2.2.1. Position Based. These crossover operators produce offspring that rely heavily on the position of certain genes in the parents. Crossover operators falling under this category are:

PMX (Partially Mapped Crossover) [Goldberg and Lingle 1985; Kumar et al. 2013; Moraglio 2007; Misevicius and Kilda 2005] – This crossover randomly chooses two cut-points and the genes between cut-points are swapped. A mapping relation is determined for the genes between the cut-points. This mapping is used to legalize the offspring as shown in Figure 6. Its variation includes the Uniform Partially Mapped Crossover (UPMX) which considers random points for swapping rather than the cut-points.

CX (Cycle) Crossover [Oliver et al. 1987; Kumar et al. 2013] – This crossover picks some cycles (subset of elements such that each element in the cycle is paired with another element in the same cycle on the alignment of the parents) from one of the two parents and the rest of the cycles from the other parent with each gene of the offspring in the same position as in either of the parents.

POSition-Based Crossover (POS) [Kumar et al. 2013] – This crossover selects a group of city's positions from one of the parents and imposes the selected positions to the cities in the other parent.

APX (Alternating Position) Crossover [Kumar et al. 2013] – This crossover creates an offspring by selecting genes alternately from both the parents, without repetition of genes.

5.1.1.2.2.2. Edge Based. Edge is the connection between two nodes. For example, the distance between two cities in TSP is indicated by the line (edge) connecting the two cities. The crossover operators producing offspring that rely heavily on the retention of certain edges of the parents in the children fall under this category. Examples of such crossover are as follows:

Edge Exchange Crossover (EXX) [Takahashi 2008] – This operator creates offspring by inheriting parents' edges to form a circuit and reversing invalid edges to create a valid offspring and at the same time inheriting all the edges in case the offspring was not able to form an entire circuit.

Edge Assembly Crossover (EAX) [Takahashi 2008] – This operator creates offspring by exchanging a set of a parent's edges with another parent's set of edges, such that the number of elements in both the sets is the same.

Heuristic Crossover [Kumar et al. 2013] – This crossover operator selects a gene at random and based on the probability assigned to its edges, using some probability distribution and cost of each of those edges, the next gene for the offspring is chosen till the entire chromosome is formed without any repetition of the genes.

Edge Crossover [Kumar et al. 2013] – This crossover operator chooses a vertex with minimum number of edges and adds it to the chromosome of the offspring as a current gene, and then deletes the gene from every other vertex's edge sets. This is recursively done till all vertices are traversed exactly once.

Edge Recombination Crossover [Whitley et al. 1991] – This crossover operator considers the chromosomes to be an undirected cycle of edges rather than the vertices considered by most permutation crossover operators. The offspring is formed by inheriting maximum edges from the parents and edges common to both parents are highly prioritized to be inherited by the offspring.

5.1.1.2.2.3. Order Based. Certain crossover operators inherit the order of genes from parent chromosomes to the offspring and hence are classified as order-based crossover operators. Certain TSP-based crossover operators fall under this category.

Sort

Sorting Crossover [Moraglio 2007] – These operators sort one of the parent's chromosomes so that it incorporates the order of the other parent's chromosome in it. According to the sort crossover algorithm, one of the parents is normalized (renumbered) based on the order of the elements in the other parent using permutation composition of a parent and the inverse permutation of the other. The minimal edit distance of transforming the normalized parent to the other is found using a sort algorithm and is considered to be the minimal distance point on the sorting trajectory. A random crossover point is chosen and the sequence of moves that were tracked is applied to the first parent from the first gene to the crossover point thereby forming the offspring.

An important point to be noted here is that not all sorting algorithms are suitable for crossover operation. Only sort algorithms like bubble, selection, and insertion sort that guarantees sorted result using minimum number of moves using the same move throughout the sorting process can be used for the sort crossover. The most efficient quick sort algorithm is not used for crossover since it does not employ the same move throughout the sort process. Examples of this type of crossover are the swap-moves-based deterministic and non-deterministic selection sort crossover (SS1X, SSUX), adjacent swap-move-based deterministic and non-deterministic bubble sort crossover (BS1X, BSUX), the insertion move based deterministic and non-deterministic

insertion sort (IS1X, ISUX) and the sorting by reversals crossover (SBRX) based on conversion distance of one parent to another.

Merge

These operators use a global precedence vector to select the preferred genes while merging the parent chromosomes to form the offspring.

Merge (1&2) Crossover [Moraglio 2007] – They form offspring based on the precedence of the genes as given in the precedence vector with the vital difference between them being that the Merge1 crossover swapping the position of a gene after it is added to the offspring and Merge2 crossover deleting them.

Merging crossover (MOX) [Mumford 2006] – Merges the parent chromosomes randomly and creates two offspring by placing the first occurrence of a gene in the first parent and the second occurrence of the gene in the second parent in the same sequence as in the merged list.

Merging Independent Sets (MIS) [Mumford 2006] – Is similar to the MOX crossover. However, a set of genes is moved to the offspring from the merged list rather than each gene separately.

Uniform Like Crossover (ULX) [Misevicius and Kilda 2005] – The precedence order is based on the like genes in the first step being transmitted to the offspring, choosing randomly from the parents in the second step by scanning from left to right of the chromosome, and filling in the remaining places in the last step. Other variations of ULX include the (i) Randomized ULX (RULX) – Where the second-level scanning is random rather than left-to-right as in ULX, (ii) Block Crossover (BX) – where the first-level composes of transmitting like genes in a block of both the parents rather than strictly in the same position as in ULX, (iii) Repair Crossover (RX) – which repairs the offspring produced by doing pairwise interchanges between genes that were not directly inherited from the same position of their parents, and (iv) Cohesive Crossover (COHX) where an initial mask is used to create offspring from parents followed by filling in the rest of the genes in a wave propagation fashion.

Relative Order

These crossover operators exploit the relative ordering of genes.

OX1, OX2, OX3, OX4, OX5 (Order Crossover) [Deep and Mebrahtu 2011] – All these operators follow a cut-point based offspring formation followed by a repair procedure that uses relative ordering of the genes in the parent chromosomes. New OX [Agarwal and Kanchan 2013] also falls under this category.

Relative Order Crossover (ROX) [Moraglio 2007] – Forms offspring by transmitting the common relative order genes of the parent chromosomes.

Uniform Order-Based Crossover (UOBX) [Mumford 2006] – Forms offspring by preserving the relative positions of the genes in the parents.

Absolute Order

These crossover operators exploit the absolute ordering of genes.

Absolute Position Geometric Crossover (APX) [Moraglio 2007] – Forms an offspring by transmitting the common absolute order genes of the parent chromosomes.

Non-Wrapping Ordered Crossover (NWOX) [Otman and Jaafar 2011] – Forms offspring by randomly creating holes in the chromosome and filling it without losing the absolute order of genes in the parents.

Adjacency

These crossover operators exploit the adjacency ordering of genes.

Adjacency Geometric Crossover (AX) [Moraglio 2007] – Forms offspring by transmitting the common adjacency order genes of the parent chromosomes.

5.1.1.2.2.4. Subset Based. This set of crossover operators work on the basis of subsets of the original set of genes of the parent chromosomes.

Subtour Exchange Crossover [Yoshikawa 2008] – Similar subsections, i.e., common set of edges called subtours of both the parents are exchanged in the offspring formation process.

Cut and Blend Crossover [Su et al. 2009] – Edges from the parents are edited to form subtours by the cut operator and blend operator blends them to form offspring.

Maximal Preservative Crossover (MPX) [Kumar et al. 2013] – This crossover operates by selecting a substring from a parent, placing it in the beginning of the offspring, and filling the rest of the offspring with the left out genes randomly.

5.1.1.2.2.5. Cut and Splice. There are a few crossover operators that cut part of the chromosome from one of the parents and splice that part into another parent to form the offspring. Normally, the part that is cut from a parent represents a solution to a constraint violation of the same part in the other parent where it is spliced into. This is more common with the timetabling problems. A chromosome in the timetabling problem is represented by 'g' genes (where each gene corresponds to an optimized variable) and each allele of a gene encodes different parameters of timetabling (like teachers, classrooms, labs, and the like). Crossover operators falling under this category are as follows:

Sector-Based Crossover [Lewis and Paechter 2004] – A sector of the timetable of a parent where constraint is violated is cut and spliced with that of the solution from another parent.

Day-Based Crossover [Lewis and Paechter 2004] – Similar to sector-based crossover. However, the constraint violation is across days.

Student-Based Crossover [Lewis and Paechter 2004] – Events of students is considered and conflicting areas of one parent are cut and spliced with that of the other parent's part.

Conflicts-Based Crossover [Lewis and Paechter 2004] – Similar to student based crossover. However, we consider the list of events conflicting with a certain event at a time.

5.1.1.2.2.6. Swap. These crossover operators perform a swap of the genes. An example of this type of crossover is the Swap Path Crossover (SPX) [Misevicius and Kilda 2005] where the common alleles between parents are left as such but the others are swapped in order to make the alleles same while performing a left-to-right scan on the parent chromosomes.

5.1.1.2.2.7. Graph Partition. These crossover operators are based on partitioning graph constructed by the union of two-parent chromosomes. While the Partition Crossover (PX) [Whitley et al. 2010] partitions the graph formed by the union of the two parent chromosomes and forms two offspring using a single partition of cost 2 (although more than one of such kind is available), the Generalized partition crossover (GPX) [Whitley et al. 2010] uses all available partitions in a single crossover to produce the offspring.

5.1.1.2.2.8. Distance Based. The distance-based crossover exploits the distance measure (number of genes in the same position of the two considered chromosomes). The Distance Preserving Crossover (DPX) [Misevicius and Kilda 2005] produces offspring which has the same distance to both the parents and this is the same as that of the distance between the parents. 'Distance' here means the number of elements present in different positions in the two-parent chromosomes considered.

The optimization of sequence-dependent problems is very challenging and hence many crossover operators were designed to suit the various types of ordering problems

as shown above. The group of OX crossover operators is the one that gives better results faster than any other crossover operator designed for TSP – a typical problem in this category, even when the number of cities is large [Deep and Mebrahtu 2011]. The MIS and the POP crossover operators work better for the graph coloring problem when the color classes vary in size and color classes are identical respectively [Mumford 2006]. From such findings, we are able to conclude that a problem-specific crossover like MIS and POP for graph coloring and sector-based and day-based crossovers for the timetabling problem perform efficiently and effectively when compared to the general crossover operators designed for permutation encoding. This is an important subclass in our classification where the application plays a major part in the type of crossover to be used rather than the representation of the application or the contribution of the crossover to the GA.

5.1.1.3. Real Encoding. Both binary and integer encoding and its variants were unable to deal with continuous search spaces of large dimensions [Deb 2000; Deep and Thakur 2007] and applications demanding high precision. Therefore, real encoding schemes and associated crossover operators suitable for such applications were designed. Real-number encoding of chromosomes bring the representation very close to the natural formulation of a huge range of problems and the GA very close to the solution space of the problem. Davis [1989] in his work explains that most real-world problems are neither efficiently represented nor effectively solved using binary strings. In the case of large problem domains, real encoding results in increased efficiency and precision [Michalewicz 1994].

The genetic process could be designed for either the phenotype (chromosome) or the genotype (Internal gene details of chromosomes). Radcliffe [1992] suggested that phenotype and genotype is not mandatory for evolution. Herrera et al. [1998] said that the schema concept and genetic operators for real coding in GAs are close to Radcliff's theory and that the expressiveness level with real encoding is very high due to the fact that the genotype and phenotype are similar here since no conversion (binary->real and real->binary) is needed in this case. Herrera et al. [1998] also argued that real coding exploits local continuities, while the binary encoding exploits discrete similarities and that real encoding is better than binary encoding for GAs. They also discuss the superiority of real encoding over binary encoding because of the ability of the former to easily integrate with domain knowledge of the problem. Ono et al. [2003] have also given guidelines on how to design a good crossover operator for real encoding. The guidelines suggest that the crossover operator should preserve the vital statistics (mean, variance, covariance) of the parental population, the offspring generated should be diverse and should enhance robustness by widening the distribution of the children when compared to the distribution of parents.

Certain applications are better represented with real encoding especially to represent the weights associated with the features. Any crossover that is designed for the real encoding scheme works well for these applications. The following applications fall under this category:

—*Animal Diet Formulation Problem* [Rahman and Ramli 2013] – This application aims at producing a diet composition with maximum nutrients and minimum cost. The weights (composition value) corresponding to each nutrient forms the alleles and hence involves real encoding.
—*Electromagnetic Optimization* [Otevrel and Raida 2007] – This problem deals with the design and optimization of the low- as well as high-frequency electromagnetic components. The weights for the components are represented as real numbers and hence real encoding suits this problem.

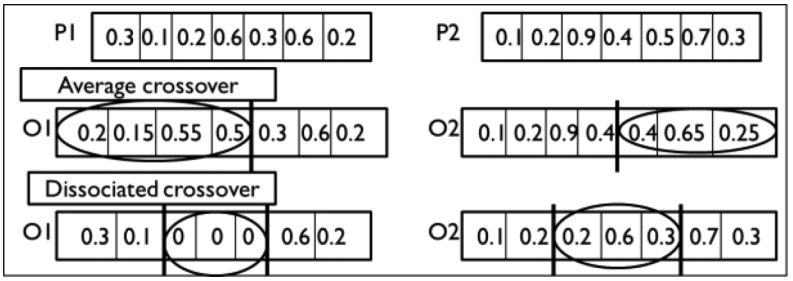

Fig. 7. Real encoding based crossover operators.

—*Revenue Management in Airlines* [Sadeghi et al. 2013] – This application deals with the allocation of airline resources which are limited in number with the aim of producing more revenue. The weights of the features form the alleles. They are in real representation and hence this problem suits the real encoding.
—*LETOR – Learning to Rank in Information Retrieval* [Vrajitoru 1998] – Considering the LETOR application, there are 134 features for each of the search documents. Weights must be allotted to the features so that the ranking function using the learnt weights gives optimal precision values. Weights use real representation and hence this application is well suited for real encoding.

The real encoding-based crossover operators can be further subclassified as follows based on the underlying operation that creates the offspring from the parents:

5.1.1.3.1. Statistical. These crossover operators use the statistics of combinations of the population like the Taguchi crossover (TC) which uses statistics of the orthogonal matrix to obtain the optimal solution even when it is absent in the orthogonal array [Chan et al. 2003].

5.1.1.3.2. Euclidean Distance Based. The sphere crossover [Grueninger and Wallace 1996] generates an average vector for the parents followed by choosing a point (offspring) within its hypersphere for which the Euclidean distance is used.

5.1.1.3.3. Arithmetic. The crossover operators falling under this category are those that form a child by choosing an allele for a gene as a combination of the allele values from both the parents. This class generally goes hand-in-hand with the real coded class. We discuss the subclasses that fall under this class in the following subsections.

Average

Some genes from one of the parents and some genes are an average of both the parents (single child). Crossover operators falling under this category are as follows:

Average Crossover [Rahman and Ramli 2013] – Part or all of the genes are averages of the same alleles in both the parents. Figure 7 shows average crossover as similar to one-point crossover. The highlighted part is the average of the genes of the same position in both the parents.

Incomplete Dominance-Based Crossover [Pavai and Geetha 2015] – Follows the incomplete dominance pattern observed in plants like the Four O'clock flowers (red and white flower-bearing plants when mated with each other, produce pink flower-bearing plants) which is similar to averaging in certain cases.

Weighted Average

A gene from a parent is combined with the same gene from a different parent with weights assigned to each gene of the parents. A crossover operator falling under this category is weighted arithmetic mean crossover – which gives different weights to the

genes belonging to different parents while creating the offspring. Operators like the Line, Xline [Yoon and Kim 2012] and NADX crossover operators [Ladkany and Trabia 2012] use this technique to generate offspring.

Upper- and Lower-Bound Based
New BLX Crossover [Morales-Ortigosa et al. 2008] calculates allele value for parents based on upper-bound and lower-bound values for the alleles.

Fuzzy Average
Fuzzy Arithmetic Weighted Mean (FAWM) crossover [Sadeghi et al. 2013] takes two parents and creates two children by computing the fuzzy weighted arithmetic mean of two parents using a fuzzy membership function (Bell shape and Triangular shape) for the fitness value calculation.

Weighted Addition
The Flat Crossover [Sharapov 2007] creates offspring by doing a weighted addition of the genes of the parents for which the weights are obtained from a pre-defined weight vector.

Center of Gravity
The Supplementary Crossover [Angelov 2001] described earlier in Section 4.1.2.1 in the binary-encoding-based crossover operators, is also applicable to real encoding.

Center of Mass
The CMX (Center of Mass Crossover) [Tsutsui and Ghosh 1998] produces offspring by calculating the center of mass between the genes of the two parents.

5.1.1.3.4. Different Operations. In most cases, a crossover operator employs the same kind of operation to create both the offspring. However, rarely, there is a chance of employing different crossover operations in a crossover.

Same – Most of the crossover operators are using the same operation for creating both the children.

Different – Two different crossover operators for forming the two children are used. The crossover operator falling under this category is the Dissociated crossover [Vrajitoru 1998]. Figure 7 shows that the dissociated crossover operator fills in 0s in part of one offspring whereas the other offspring is created in a fashion similar to that of the two-point crossover operator.

Real coded crossover operators generate offspring with higher diversity. Consider the averaging and the incomplete dominance-based crossover where, a part of the offspring chromosome has genes that may not be present in the generation of their parents. However, the same reason makes the real coded crossover operators lose the good genes of their ancestors [Mudaliar and Modi 2013]. Moreover, real encoding suffers from the local optima problem more than any other encoding because of diversifying the search rather than concentrating on a particular area of the search space that is close to the global optima. This provides a very good ground for future research on real coded crossover operators.

5.1.1.4. String Encoding. One of the important applications of GA was in the area of bioinformatics that required a natural and easy way to represent features in text-based applications. The DNA sequencing [Moraglio 2007] problem basically requires representing the proteins of the DNA which are by nature character strings thereby making this application use string encoding. Crossover operators to handle these string-encoded chromosomes are apt for this purpose.

This subclass under the representation-based classification of crossover operators further subclassifies the crossover operators based on whether the length of the chromosomes that are used to represent the individuals are fixed or variable. Two parents

have equal number of genes in the case of fixed length and unequal number of genes in the case of variable length.

5.1.1.4.1. Fixed Length Representation. Length of the chromosomes of the individuals in a population is fixed for many applications making the crossover operation less complicated. The representation is straightforward in this case and the job of the crossover operators while producing offspring enables the passing of either parents' allele (or a combination of the parents' allele) to the corresponding gene of the offspring. This eliminates the question of deciding which gene to add and which gene to delete from parents to create offspring.

C1 Crossover [Jackson 2004] – This crossover selects the cut-points based on preserving the secondary structure element of the DNA sequence.

C2 Crossover [Jackson 2004] – Selects the cut-points based on the frequency of pairs of secondary structures appearing adjacent to each other.

The fixed length representation is the most commonly used representation owing to its simplicity. The crossover operators discussed so far belong to this class. However, this type of representation is not flexible and hence will only suit applications where the solutions are better represented in fixed length form.

5.1.1.4.2. Variable Length Representation. Applications like biological sequencing where DNA, RNA, or peptide sequence is analyzed to find the underlying features, function, structure, and the like, are better represented with variable length chromosomes since this form of representation is close to their natural representation. In this case, while creating the offspring, it is generally difficult to decide on which gene is to be added and which is to be deleted and from which parent. The number of genes could be same as that of parent1 or that of parent2 or an average of the lengths of parent1 and parent2 or a combination of all the genes of both the parents (if so, there is a question on the order of the genes in the offspring). Baake [2008] called such crossover operators *unequal crossover operators*. Stringer and Wu [2005] have classified variable-length GAs into (1) Fixed size solution that requires deciding the number of variables and their instantiation beforehand, (2) Bounded size that requires stating maximum number of possible variables without the need for instantiation for completeness, and (3) Unbounded size solutions for complex problems.

Crossover operators designed for unequal chromosome length population are as follows:

Internal Crossover [Baake 2008] – This crossover operator works on the basis of the assumption that the shorter of the two chromosomes can align with any connected block of the longer chromosome without overhang. This operator is also called as the *Internal Unequal Crossover*.

Random Crossover [Baake 2008] – This crossover operator allows arbitrary overhangs, such that one offspring sequence gets all the genes under consideration while the other sequence loses everything.

In Crossover [Stringer and Wu 2005] – The offspring produced using this operator are such that both offspring are in between both the parents in their length (inside event) as shown in Figure 8.

Out Crossover [Stringer and Wu 2005] – The offspring produced using the out crossover are such that both children are less than and greater than the size of both the parents (outside event) as shown in Figure 8.

Equal Crossover [Stringer and Wu 2005] – The offspring produced using the equal crossover are such that children are of a size equal to the parents (equal event) as shown in Figure 8.

Variable-length crossover operators cater to flexible representation of the applications especially in the case of string alleles. However, certain applications are better

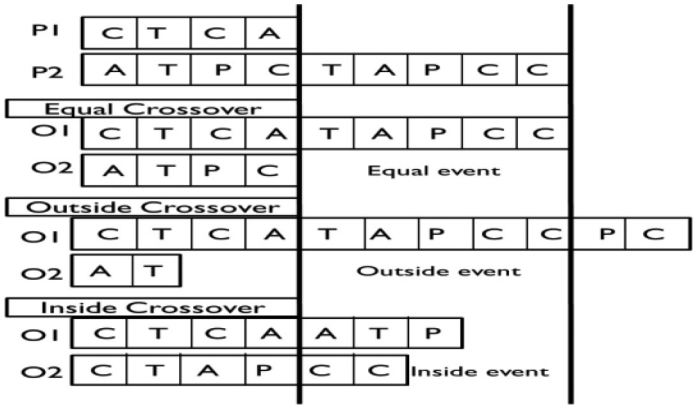

Fig. 8. Variable length based crossover operators.

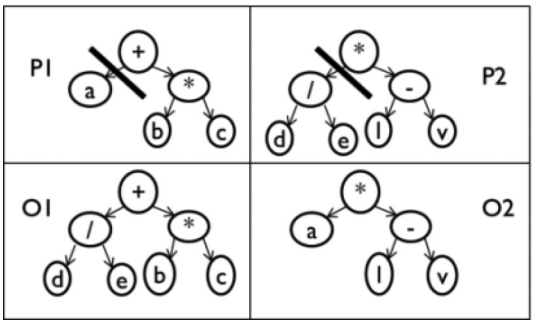

Fig. 9. Tree-encoding-based crossover operator.

represented in the non-linear form rather than the linear form described here. This helped in the design of many multi-dimensional crossover operators that are described in the following section.

5.1.2. Crossover for Multi-Dimensional (Non-Linear) Representation. A poor representation limits the power of GA, and sometimes is responsible for obtaining a false solution. This was the driving force for research in the area of new encoding schemes for GAs. Radcliffe [1992] has suggested that for many problems conventional linear chromosomes and crossover operators are not very effective and that non-linear representations are required. We need representations which help to define operators which directly manipulate solutions in "sensible" ways [Radcliffe 1992]. Apart from this, Radcliffe [1992] argues that representation independence and inappropriate schemata support the non-linear case. Non-linear representations need to be used when conventional schemata cannot clearly capture the desired characteristics or constraints of the problem and the solution space. Some of the non-linear representations are given below along with reasons that support that kind of representation for an individual in the population, the example crossover operators that use that type of encoding and applications where that type of encoding can be used. Generally non-linear encoding is considered to be a better representation than the linear representation owing to the former's increased versatility. However, this is always at the cost of space.

5.1.2.1. Tree Encoding. An individual is a string represented by trees in this case as shown in Figure 9. This representation requires more space but is highly efficient. A

fixed-length encoding without variation is artificial and, such encoding schemes cause redundancy and reduces the efficiency of genetic search. Trees use high-level building blocks of variable length. Moreover, their size and complexity has the power to vary during breeding. The tree encoding is the most commonly used encoding in the non-linear case which is evident from the fact that a class of GAs that use tree structures for representation is called Genetic Programming (GP) [Koza 1992]. The following applications are better represented with tree encoding:

—*Syntactic Trees* [Moraglio 2007] – Syntactic trees represent the syntax structure of the program codes. Considering the Syntactic trees, they are naturally and easily represented in tree format thereby demanding the design of the solutions in tree format as well and hence demands the use of tree representation for these kind of problems.
—*Two Boxes* [Heywood and Zincir-Heywood 2002] – This problem consists of a person predicting the presence of valuables in one of the two mystery boxes, where the set of states leading to the conclusion is represented in tree format.
—*Parity* [Heywood and Zincir-Heywood 2002] – The parity problem considered here is even parity where the number of bits with a '1' value are counted. In case it totals up to an odd number, the parity bit becomes '0', the total number of '1's becomes even. When the count is an even number, the parity bit does not change and maintains a '0' value. The edges of the parity tree hold the binary values and the leaves of the tree store the parity value for each path from the root to that leaf.
—*Classification* [Heywood and Zincir-Heywood 2002] – This problem classifies a new observation as belonging to some predefined class based on patterns derived from similar observations and the features and their values leading to each class of the classification being represented in the form of a tree.
—*Phylogenetic Inferencing* [Matsuda 1996; Bates Congdon 2002; Lewis 1998; Reijmers et al. 1999] – Phylogenetic inferencing uses molecular sequential data or morphological character data with an aim to classify organisms into a taxonomy, giving special consideration to relationships between them.
—*Maximum Parsimony Problem* [Goeffon et al. 2006] – This problem is solved by the reconstruction of a phylogenetic tree from the DNA sequences with minimal evolutionary changes and hence naturally suit the tree representation.

The basic crossover operators can be used in the case of tree encoding as well. Other existing tree encoding based crossover operators can be further subclassified as follows based on the underlying operation responsible for creating the offspring from the parents.

5.1.2.1.1. Swap. These crossover operators perform a swap of the subtrees of the parents. While the Homologous crossover [Langdon and Poli 2002; Moraglio 2007] swaps a subtree whose node falls in the boundary of common region between the parents, the One-point tree crossover [Moraglio 2007] swaps the subtrees at the selected cut-points and the Sub Tree Swap crossover [Koza 1992; Moraglio 2007] swaps any subtree of a parent with any subtree of another parent as shown in Figure 9 (here, subtree is rooted at the node following the cut-point of each parent). Page-Based and Dynamic Page-Based Crossover Operators [Heywood and Zincir-Heywood 2002] also perform the swap of pages with the Page-Based Crossover allowing an initially fixed page size and the Dynamic Page-Based Crossover allows dynamic change of the page size, however, within the same individual, the overall size must be maintained.

5.1.2.1.2. Semantic. Semantic crossover operators are specific to the semantic Genetic Programming (SGP), which not only considers fitness as an important criterion but also the information acquired from the program behavior called semantics which is used to

construct search operators for improving the search results. The crossover operators that fall under this category are as follows:

Krawiec and Lichocki Geometric Crossover [Pawlak et al. 2014] – The best two offspring from N offspring generated using N crossover operations to the same set of parents based on the degree of geometricity such that the chosen offspring is of minimum distance from the parents and at the same time not the clones of the parents (with the help of semantics from parents).

Exact Semantic Geometric Crossover (SGX) [Pawlak et al. 2014] – Merging both the parents along with their randomly created offspring to form a new offspring such that the semantics helps the offspring fall geometrically between the parents.

Locally Geometric Semantic Crossover (LGX) [Pawlak et al. 2014] – The cut-point in common region between the parents is chosen and intermediate offspring are created with this as the root followed by calculating the semantics such that it is the midpoint between the semantics of the created offspring. An existing program 'library' is referred to for a program that is similar to the obtained semantics which forms the actual offspring.

Approximately Geometric Semantic Crossover (AGX) [Pawlak et al. 2014] – The calculation of semantics for this crossover operator is the same as it was given for the LGX crossover. Semantic back-propogation of semantics is done through parent trees from their root to the selected crossover point followed by choosing a similar program from the program library as the offspring.

5.1.2.1.3. Cut and merge. The cut and merge operation is done for crossover where portions of the parent trees are cut and are merged based on certain criteria.

The Matsuda [1996] crossover cuts the common genes containing minimum subtree of the parents and a merge-point-rooted subtree from a parent and merges with the rest of the nodes in the other parent.

The Congdon [2002] crossover cuts a subtree from a parent and inserts it into the place of a minimum subtree consisting of all nodes in the other parent.

The Lewis [1998] crossover cuts a subtree from a parent and all the other nodes from the other parent and merges the remaining parts of both the parents.

5.1.2.1.4. Matrix Addition. These crossover operators employ matrix addition to form offspring as in the case of the Reijimer's [Reijmers et al. 1999] crossover where a distance matrix (which has the distance between nodes) and a correction matrix (which has correction distances) are added. Distance-based Information Preservation (DiBIP) crossover [Goeffon et al. 2006] also uses matrix addition where the matrix is represented as the semantic information between nodes of a phylogenetic tree.

5.1.2.1.5. Distance Based. These crossover operators work based on the distance between nodes. The Deterministic Functional Crossover Operator (D-FXO) [Bongard 2010] uses the behavioral distance rather than the structural distance between the nodes to form offspring.

5.1.2.1.6. Probability Based. These crossover operators work based on the calculation of the probability that a node is chosen as a cut-point like in the Probabilistic Functional Crossover Operator (P-FXO). After the cut-points are chosen based on the calculated probability, swapping is done to form offspring.

The tree encoding is the most common non-linear representation and the fact that many crossover operators are designed for this encoding speaks about its popularity in the non-linear case. However, tree encoding works well for smaller tree sizes. Any known crossover is not pointed out in the literature to handle the larger trees efficiently and effectively.

5.1.2.2. Matrix Encoding. An individual is a matrix rather than a linear data structure or tree-shaped one. There is an increased efficiency in GA for problems using this representation. Any application that has the parameter space in the form of relation between two classes is suitable to be represented in the matrix form. The Reijmers Crossover operator [Reijmers et al. 1999] works for chromosomes with matrix encoding. This operator represents the trees as distance matrices that describe the distance between nodes of trees. This matrix is called as the correction matrix whereas another randomly generated 'correction matrix' is evolved to obtain an optimized working matrix.

5.1.2.3. Array Encoding. Some crossover operators are designed to work on array encoding. The individuals are represented as arrays in this type of encoding. Some of the array based crossover operators include the Orthogonal array based crossover operators [Picek et al. 2011] representing the interactions between all the genes in array form (which is used for forming the offspring).

5.1.2.4. Module Encoding. This kind of representation was first designed for the Software modularization problem. This is not a very simple problem since we need to identify each possible way of modularizing the system under consideration in a unique way so that we have only one representation per every modularization. In case the modularizations are non-unique, there is an increase in the search space size, which limits the use of any kind of search-based approach to the problem. Harman et al. [2002] have described the module based representation as numbering the modules and allocating elements to them with a look-up table. This representation suffers from the need to renormalize when components are rearranged as the effect of crossover and mutation. This method has the advantage that a particular allocation of components to modules has only one single representation. Novel crossover operator for module representation [Harman et al. 2002] chooses a parent of the two mating parents arbitrarily and copies one of its modules to the child and removes the module from both the parents (for duplicate module removal). This copying process is repeated till the child contains all modules and the parents have finished with all their modules.

5.1.3. Crossover Operators for Applications Using Specially Designed Representations. Apart from the above-mentioned encoding schemes, there are a few more encoding-schemes-like expressions, in case of applications like control tasks, robotic planning, and symbolic regression [Koza 1992], hybrid encoding which is a combination of binary (for discrete domains) and real encoding (for continuous domains) [Sharma and Irwing 2003], Fuzzy coding "where every attribute has a fuzzy set and a set of degrees of membership to each fuzzy set" [Aguilar-Ruiz et al. 2007], Binary + Permutation encoding where each gene is represented by its contribution to the fitness value of the chromosome instead of the binary encoding [Kumar and Jyotishree 2012] and Natural encoding where the continuous and discrete attributes are converted to natural numbers [Aguilar-Ruiz et al. 2007]. While most of the above representations adapt crossover operators that are already defined for some other encoding, natural encoding has a specially designed crossover operator called the *natural crossover operator* which produces an offspring from two parents that underwent mutation where the values of the offspring are from the first non-empty intersecting point between the parents [Aguilar-Ruiz et al. 2007]. In RGF (Restricted Growth Function) encoding [Swift et al. 2007], the discrete uniform distribution suggested by the restricted growth function gives alleles to offspring. A corresponding RGFGA crossover operator [Swift et al. 2007] is defined on the basis of choosing two offspring at random from the path defined by the Hamming distance between the parents. Context-sensitive crossover operator [Banerjee

and Dave 2010] is also defined for RGF encoding which creates an offspring that inherits best genes for each cluster from either of the parents. Cartesian Genetic Programming [Miller and Thomson 2000], a variation of the traditional GP differs from the later by the usage of directed graph representation given as a list of integers rather than the tree representation. Clegg et al. [2007] have used a slight variation of the above representation by using a list of real numbers rather than integers to represent the directed graph. Both the above works [Miller and Thomson 2000; Clegg et al. 2007] have used crossover operators designed to suit the respective representations.

5.2. Crossover for Multiple Representations

Certain crossover operators are not restricted to work for only one type of representation and can be adapted conceptually to operate on more than one kind of representation. For example, the basic one-point crossover operator can be used for integer encoding as well as real encoding (and other encodings). The following subclassifications are made based on the operation that creates offspring from the parents.

5.2.1. Cut-Point Based. These kinds of crossover operators are responsible for passing on the same set of genes across generations thereby restricting the search space by leaving out certain string of better genes if the overall fitness of the individual containing the string was less and hence discarded. The one-point, two-point and the n-point crossover operators described earlier fall into this category. Other crossover operators falling under this category are:

M-X (M) Crossover [Mudaliar and Modi 2013] – This crossover operator produces eighteen children from two parents in a manner similar to the two-point crossover. Three blocks of genes are produced for each parent (produced by the two cut points). Each block is placed before a block in the other parent and then the duplicate genes are removed thus producing eighteen children for two parents.

Reduced Surrogate Crossover [Picek et al. 2011] – This operator reduces parents to a skeletal form where bits that differ in two parents alone are represented followed by Single-point crossover to form children.

Dissociated Crossover [Vrajitoru 1998] – This operator employs two different crossover operations to produce the two offspring in a process that is similar to the two-point crossover.

Ring Crossover [Kaya et al. 2011] – This operator was designed for the binary encoding where the offspring are generated by following a ring order rather than the linear order in two-point crossover operator.

Sorted-Match Crossover [Kumar et al. 2013] – This crossover operator creates an offspring by choosing a parent with a high cost subtour and exchanges the subtour with a low cost subtour from another parent where the subtours chosen should start and end with the same genes (cities) and contain the same set of genes in between them.

Partheno Crossover (PCGA) [Jin and Zhang 2007] – This crossover operator performs single-point crossover N times with a different locus each time to form $2N$ offspring and the best offspring replaces the parent.

Permutation One-Point Crossover (POP) [Mumford 2006] – This crossover operator is close to the one-point crossover. However, the genes after the cut-points in both the offspring are filled with genes that are not already present and follow its order in the parent. Two variations of POP crossover called POP1 (where cut-point is chosen at random) and POP2 (where the cut-point is normally a border that appears in between two color classes.

Though the underlying principle of the above-mentioned crossover operators is cut-point based, the problem-specific elements like ordering to be maintained for

POP crossover is very important for the design of these crossovers. Cut-point-based crossovers are generally very simple and hence easy to use since there are no extra efforts involved. However, they do not show a good performance, since cut-point-based crossover operators generally tend to rotate the same set of alleles from generation to generation.

5.2.2. Random Order. Random order of genes from each parents are chosen each time. They reduce the chances of circulating the same set of genes in the population since each gene is selected randomly from a parent rather than choosing a set of genes together from a parent. The Uniform crossover operator described earlier is a very good example of the random-order-based crossover operators with the randomness playing a key part in selecting alleles for the offspring. Crossover operators falling under this category are:

Distance Preserving Crossover (DPX) [Misevicius and Kilda 2005] – This crossover operator produces offspring having the same distance from both the parents which is in turn equal to the distance between the parents where distance is the number of elements assigned to different positions in the chromosomes. It works by creating an offspring by filling in the positions with the same elements in both the parents and then by filling in the other positions with the remaining genes in a random order.

Box and Xbox (Extended Box) Crossover [Yoon and Kim 2012] – These operators produce offspring based on random selection of genes from their parents.

Random-order crossover helps to lose bad genes due to the random process. However, the information obtained previously through GA operators is destroyed due to the crossover operator's random choices of crossover points.

5.2.3. Compound. They combine two or more crossover operators so as to get the good properties of the combined operators. Examples of compound crossover operators are:

Uniform Wise Crossover (UWX) [Otevrel and Raida 2007] – This is a combination of the UX (Uniform crossover) and PWX (Parameter Wise Crossover – which is a multi-parent crossover described in Section 4.2.2.2).

Product Geometric Crossover [Moraglio 2007] – The geometric framework can formulate the Cartesian product of geometric crossovers, which helps to build new geometric crossovers from existing geometric crossovers. There are various types of product geometric crossover operators like multi-crossover [Moraglio 2007] which forms offspring using of the same crossover with the same representation as many as N times, Hybrid crossover [Moraglio 2007] that forms offspring using of different crossovers with the same representation for each of the different kinds of projection, Hybrid representation crossover [Moraglio 2007] which forms offspring using of different crossovers with different representations for each of the different kinds of projection, Dependent crossover [Moraglio 2007] where the projections represent single entities and are mutually constrained.

Quotient Geometric Crossover Operators [Moraglio 2007] – This crossover is similar to the product-geometric crossover other than using quotient-metric spaces rather than the product-metric spaces of the latter. Labeling-independent crossover [Moraglio 2007] is a quotient geometric crossover operator that normalizes the second parent to the first under hamming distance and then normal crossover is done.

Cycle LI-GX Crossover Operators [Moraglio 2007] – This crossover operator does labeling-normalization of a parent before applying the cycle crossover.

PMX+OX [Kumar et al. 2013] – is a combination of the PMX and the OX crossover operators described earlier.

The compound crossover operators increase the explorative power of GAs since the different crossovers contribute to different amounts of exploration and the sum of the explorative power of these different crossover operators are obtained by the GA in

the case of compound crossover operators. Furthermore, compound crossover helps in getting better results faster.

5.2.4. Heuristic. A gene that has a higher chance of increasing the fitness of an individual is chosen for the child based on some heuristics. These types of crossover operators generally produce only a single child.

Wright's Heuristic Crossover [Wright 1991; Herrera et al. 1996] – This crossover operator uses the fitness value to determine the direction of search, i.e., in forming offspring. This crossover operator produces a single offspring that lies in the exploration zone of the best parent.

Linear BGA Crossover [Schlierkamp-Voosen 1994; Herrera et al. 1996] – Uses the heuristics to form offspring better than the parents. This crossover operator can produce the offspring in both the exploration and exploitation zones.

Dominated Crossover [Herrera et al. 1996] – Uses the heuristics of prioritizing the transfer of the best parent's genes to form offspring better than the parents by combining the effects of F and S crossovers.

Biased Crossover [Herrera et al. 1996] – Forms an offspring from two parents as a result of heuristic selection of genes from the parents.

Intelligent Crossover (IC) [Ho et al. 2002] – Estimates every gene's contribution to the fitness of the individual using the reasoning ability of orthogonal experiment design thereby selecting better genes to form the offspring.

Heuristic Crossover [Poli and McPhee 2000] – This crossover operator being designed for the TSP problem, considers edges along with the vertices (rather than either of these as in most crossover operators) and the heuristic in the form of distance between the cities is considered for creating an optimal offspring.

Adaptive Crossover [Korejo et al. 2013] – Uses reward functions as a heuristic based on the fitness values of parents and the offspring they produce to form better offspring.

Heuristic Weighted Crossover (HWCO) [Peng and Wang 2007] – Concentrates the search operation in the search space close to the better parent which heuristically gives better offspring.

Rough Set Theory-Based Crossover Operator (RSO) [Li et al. 2005] – Uses attribute reduction method in rough set theory to find genes that are likely to show good performance.

Gene Memory-Based Crossover [Pavai and Geetha 2015] – The best allele value for each gene is recorded at every generation so that the best alleles do not get lost in the process of evolution. The heuristics that all best alleles form the chromosome gives the best fitness.

The heuristic crossover operators pass on only high fitness regions of the solution space to the offspring which leads to quicker convergence and better solutions.

5.2.5. Hybrid. These are crossover operators where more than one operator (belonging to the homogeneous category) is used to produce offspring from the same population. This category of crossover operators is similar to the compound category that was discussed in the previous section as both these classes of crossover operators use more than one crossover operator for producing offspring. However, the main difference between the two classes is that the compound class combines two or more crossover operators as two steps of a single crossover operation. Whereas, hybrid class has more than one crossover operators working independent of each other to produce offspring for a set of parents, and the offspring with best fitness value is chosen without considering which crossover operator produced the offspring. The following are hybrid crossover operators:

Max-Min-Arithmetical Crossover (MMAX) [Herrera et al. 2003] – This crossover operator creates two offspring by means of any discrete crossover and another by means of arithmetic crossover.

HST1 Crossover [Herrera et al. 1996] – This operator creates two offspring: one from dominated crossover and another with biased crossover.

Fuzzy Connectives-Based Crossover (FCB) [Herrera et al. 1997] – One of the operators F, S, M and L is randomly chosen to create offspring for a set of parents.

The hybrid crossover operators avoid premature convergence problem (solution is obtained faster but converges to a local optimum rather than global optimum) since they keep the exploration/exploitation levels correctly so as to maintain diversity in the population due to the fact that offspring are produced by different mechanisms.

5.2.6. Neighborhood Based. These are crossover operators that determine the genes of the offspring by extracting values from intervals that are defined by neighborhoods associated with the genes of the parents based on a probability distribution. Examples of neighborhood-based crossover operators are:

—BLX-α crossover based on uniform probability distribution [Herrera et al. 2003; Sharapov 2007].
—Simulated Binary Crossover (SBX) is based on exponential probability distribution [Deb and Beyer 1999; Herrera et al. 2003].
—Fuzzy Rule based on triangular probability distribution [Herrera et al. 2003].
—*Guo Tao's crossover (GTX)* – Based on linear non-convex combination of n-parent chromosomes [Li et al. 2003].
—*Adaptive Neighborhood-Based Multi-Parent Crossover (ANMX)* – Based on linear non-convex combination of n-parent chromosomes where the selection of alleles for the offspring is uniform [Li et al. 2003].

Crossover operators other than neighborhood-based shows a similarity in producing the same offspring as a result of crossing over the same set of parents (deterministic behavior), whereas neighborhood based crossover operators do not produce the same offspring for the same set of parents. This helps in exploring a wider area of the search space. We describe the crossover operators for improving GA performance of applications in the next section.

6. CROSSOVER FOR IMPROVING GA PERFORMANCE OF APPLICATIONS

Some common issues in GAs are the inability to achieve optimal solution, the increased time to converge to an optimal solution, getting into local optima and premature convergence. An interesting aspect is that all these performance issues of GA can be related to crossover operators. Some crossover operators were developed to handle these issues. In Section 5, the relationship between the crossover and particular applications was discussed since representation is directly dependent on the application. However, in this section, we discuss the types of application as a group like uni-modal and multi-modal, rather than specific applications since GA performance is considered to behave in the same manner for groups of applications. In this section of the classification, we describe such crossover operators that were designed to improve the fitness and speed of GA.

6.1. Crossover for Improving Quality of GA Solutions

The most important problem with GA is its inability to converge to global optima when the search space is huge leading to poor quality (fitness values) of the solutions. As we had mentioned earlier, crossover being an explorative force helps in exploring powerful areas of the search space. However, specialized crossover design for handling this issue has to be concentrated on. In this section, we describe some of the crossover approaches designed to tackle the problem of improving the quality of GA solutions with respect to fitness of applications.

6.1.1. Applications and Adaptive Crossover. Adaptation has gained a high level of importance for enhancing GA performance. The adaptive techniques dynamically acquire knowledge on the search space and adjust the performance of GA according to the acquired knowledge. Adaptation finds its importance in complex applications like protein folding and peptide docking of biological sequencing (these are applications where a set of strings are recombined to form new set of strings), where global optimum needs to be found based on the dynamically obtained new proteins and peptides at every new generation.

Vekaria and Clack [1999] have discussed some of the adaptive crossover operators in their work. They are (1) Punctuated Crossover, which evolved the positions at which the crossover should occur with the help of crossover bitmaps appended to the end of the obtained solutions, (2) Masked Crossover, which uses binary masks to direct crossover where the fitness information about the solutions are represented as binary masks, (3) Adaptive Uniform Crossover, which augments each bit string in the population at each bit position by an automaton, (4) 1-bit adaptation that allows GAs to choose between uniform and two-point crossover, and (5) Selective Crossover that biases alleles that has previously increased an individual's fitness using a real-valued vector that represents the fitness information.

Yang [2002] described three levels of adaptation in crossover operators with the crossover operators themselves being adapted at the top level, the rate or crossover probability being adapted at the middle level and the position of crossing over or swapping being adapted at the bottom level and there are crossover operators associated with these levels of adaptation.

Korejo et al. [2013] proposed an adaptive crossover that modifies the probabilities of crossover operators based on the existing generation's global behavior for getting the new generation.

Adaptive crossover forms an integral part of the future of GAs since the prediction of a better crossover rate or a better crossover operator for the results based on various factors from the previous generation is very important for certain applications. This can be considered a major area for future work on crossover operators.

6.1.2. Applications and Multi-Parent Crossover. Crossover operators can vary with the number of parents that are considered for reproduction. This class can be further subdivided based on whether the number of parents for a crossover operation is two or greater. The two-parent case is the most common one as it mimics the natural evolution process. Most crossover operators fall under this category. More than two parents producing an offspring do not exist in nature. However, this method was developed with the intention of extracting better genes from more than two parents in order to get better child(ren). It is empirically proved that as the number of parents increases, performance improves in many cases in terms of the quality of the converged solution [Tsutsui and Ghosh 1998]. In general, multi-parent crossover operators work well on applications with multi-modality (search spaces with multiple peaks), and/or epistasis (influence of a gene over others) since genes from multiple parents are considered for offspring formation. Based on the order in which the multiple parents are considered for the offspring formation process and what the offspring base is, this class can be further subdivided into the subclasses 'order of parents' and 'offspring base' which are described in the following subsections.

6.1.2.1. Order of Parents. Since more than two parents are involved in the offspring formation process, the order in which the parents are considered forms an important aspect of classification of the multi-parent crossover operators. The recombination of parents usually takes place in one of the following two forms:

Sequential Consideration – Where the parents are considered one or two at a time and form intermediate children followed by consideration of the next set of parents for creating the next set of intermediate children and so on till a specific number of parents have been considered.

Simultaneous Consideration – Where all parents are considered together (some genes from some parents) to form offspring.

6.1.2.2. Offspring Base. Apart from the above classification of multi-parent crossover operators, we can also classify the multi-parent crossover operators based on whether the offspring is formed with a parent as a base or a template of the chromosomes as a base.

Parent as a Base – The crossover operators operating by considering one of the parents as the base for the offspring to be generated and replacing certain genes positions of this parent with genes from the other parents fall under this category. Examples of such crossover operators are as follows:

Center of Mass Crossover (CMX) [Tsutsui and Ghosh 1998] - This operator produces m children from the center of mass of N randomly chosen parents.

Multi-Parent Feature-Wise Crossover (MFX) [Tsutsui and Ghosh 1998] – This operator generates N offspring from a set of N randomly chosen parents using different feature wise information of the m parents.

Seed Crossover (SX) [Tsutsui and Ghosh 1998] – This operator generates N offspring from N parents by arranging the selected parents in the ascending order of their ranks (generated by the fitness values).

Template of Chromosome as a Base – The crossover operators use a template (size and encoding) of the chromosomes in the population in this case. The gene values are initially unfilled unlike the previous case where the gene values are that of the parent considered as the base. Examples of such crossover operators are as follows:

Scanning Crossover [Eiben et al. 1994] – This method produces a child from $n > 1$ parents by scanning the child gene positions from left to right using marking strategies to decide on which parent will contribute the current gene to the offspring to be produced. There are other types of scanning crossover like the (1) Uniform scanning, (2) Occurrence-based scanning, (3) Fitness-based scanning, and (4) Adjacency-based scanning crossover.

Diagonal Crossover [Eiben et al. 1994] – It is equivalent to the N-point crossover but for multiple parents forming offspring when, $N \geq 1$ produces $N + 1$ offspring from $N+$ 1 parents.

Multiple Parent Crossover (MPX) [Misevicius and Kilda 2005] – The number of times an element appears in position j in N parents decides the gene for a position of the offspring.

Uni-modal Normal Distribution Crossover (UNDX) [Ono et al. 2003] – The UNDX crossover uses normal distribution defined by three parents for creating children around their center of mass.

SPX (Simplex Crossover) [Tsutsui and Ghosh 1998] – This operator works on N parents to produce N children using the simplex of the search space.

Parameter-Wise Crossover (PWX) [Otevrel and Raida 2007] – It creates offspring by alternately choosing genes of all the parents.

Triadic Crossover [Pal 1994] – This operator works by creating an offspring by filling the gene positions of two parents where their values are equal and the remaining gene positions with the values of the corresponding gene positions from the third parent.

VR (Voting Recombination) Crossover [Kumar et al. 2013] – This crossover operator selects a gene for an offspring by checking if the gene occurrence in the N-parents has crossed a fixed threshold. For the rest of the genes (not filled by the threshold – voting

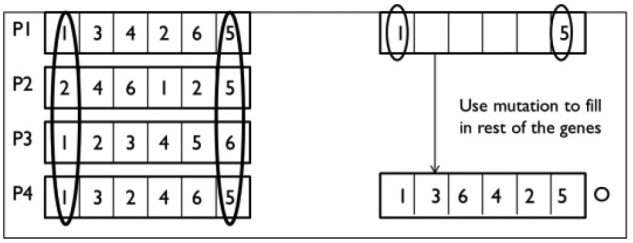

Fig. 10. Multiple parent crossover (VR).

scheme), mutation is used. This is clearly explained in Figure 10 where number of parents is 4 and the threshold is 3 (genes crossing the threshold are highlighted).

Choosing the number of parents for multi-parent crossover is very important since the more the number of parents, the higher is the exploration area which leads to (1) an improved rate at which the global optimum is found (increased effectiveness) and (2) an increase in the computation cost (decreased efficiency) thereby triggering the effectiveness vs. efficiency tradeoff.

6.1.3. Applications and Single and Multi-offspring Crossover. This class divides the crossover operators based on the number of offspring produced as a result of the crossover operation. This class can be further sub divided into the single, two and multi-offspring classes. The two offspring case is normal in GAs and many crossover operators fall under this category. The more natural single offspring crossover along with multiple offspring crossover is discussed below:

Single Offspring

One child, which is better than both the parents, is formed by crossover operators belonging to this category. All crossover operators that produce single child are aiming at producing a better child than the parents. Crossover operators falling under this category are as follows:

—*Fusion Crossover* [Beasley and Chu 1996] – This operator employs heuristics of transferring genes that contribute to a better fitness of the parents to the offspring.
—*Heuristic Crossover* [Kaya et al. 2011] – This operator produces one offspring that is better than both of its parents such that the offspring lies closer to the parent with a better fitness and away from the parent with less fitness.
—*Biased Crossover* [Herrera et al. 1996] – This crossover operator forms an offspring from two parents as a result of heuristic selection of genes from the parents.

Multiple Offspring

A few crossover operators fall under this category. Some crossover operators produce more children and choose the best among them. Herrera et al. [2003] have shown in their work that this strategy improves the behavior of RCGAs. Crossover operators that create more than two offspring are given below:

—Most operators, like the LX Crossover [Herrera et al. 2003], MMAX Crossover [Herrera et al. 2003], and the Optimized Crossover [Nazif and Lee 2010] and others, produce many children and choose two best offspring (one best offspring in the case of the Optimized Crossover operator which is used to form the other child).
—DST1 and DST2 [Herrera et al. 1996] are crossover operators that form four children for a pair of chromosomes and chose the two best in the former case while all the four are selected in the latter case.

—Most of the Multi-Parent Crossover operators like CMX, MFX, SX, Simplex [Tsutsui and Ghosh 1998], Scanning [Eiben et al. 1994], and the Diagonal Crossover operators produce as many children as the number of parents.

The children are formed merely for expanding the search space. There is no guarantee that the children so formed will be better than the parents. Better children from the so-formed set can be chosen for the next generation. However, forming a single better offspring and choosing a better offspring out of the many formed aims at concentrating on better regions of the search space. The multi-offspring strategy is more fruitful when used for applications where the search space is continuous and very large. Multiple offspring are formed and the potential offspring are alone chosen leaving behind the unnecessary offspring. This helps in narrowing down the search space to potential areas after examination rather than just blindly following one direction in the search space. The single offspring case can also be considered in applications that have a large search space. A single potential offspring is created in this case against producing many offspring and then checking for its potential as in the case of multi-offspring crossover. However, it is very highly challenging to create offspring that are definitely fit compared to the parents and this involves an intelligent design of the crossover operator so that the time and effort is not wasted in generating less fit intermediate offspring like what some of the existing crossover operators do. We discuss on the crossover operators for improving GA speed in the next section.

6.2. Crossover for Improving GA Speed

GAs generally takes a long time to converge to an optimal solution. This led to the design of new techniques to help GA converge faster. One such important technique for improving the GA speed is the crossover operators. However, improving the speed brought about a serious defect to GA which is deterioration in the quality of the converged solution due to cutting short of the search space as a result of limiting the time for search. This is a new class of limitation in GA and is termed as the 'premature convergence' problem. Research on reducing the premature convergence in GAs brought about many techniques. The crossover operators designed for handling this important limitation of GA is discussed in the forthcoming sections.

6.2.1. Faster Convergence. This section discusses some of the crossover operators designed exclusively for faster convergence. The problem with faster convergence is that it adversely affects the quality of the converged solution. However, the following crossovers claim that they help in quicker convergence without affecting the quality of the solution.

Rank and Proximity Based Crossover (RPBC) [Chakraborty and Hoshi 1999] – This crossover operator forms offspring from chromosomes of similar ranks and from closer locations. This causes disruption of good chromosomes to be avoided. This in turn helps in quicker convergence to the solution.

NEW Crossover Operator – This crossover operator [Lunacek et al. 2006] preserves the valid bit combinations that help in utilizing the sparse area having useful solutions in the search space. This concentration on the useful area of the solution space brought about quicker convergence to the solution.

MPX, MLX (Multi-Parent Crossover Operators) – Multi-Parent Crossover operators described in Section 6.1.2 though designed for obtaining good quality solutions also help in faster convergence since features of more than two parents are considered at a single shot for developing the offspring. MPX (multi-parent crossover with polynomial distribution) and MLX (multi-parent cross-over with lognormal distribution) are two such multi-parent crossover operators [Patel and Raghuwanshi 2011] designed for multi-objective optimization.

New OX [Agarwal and Singh 2013], PMX +OX [Kumar et al. 2013], M-X [Mudaliar and Modi 2013], LGX [Pawlak et al. 2014], RGFGA [Swift et al. 2007], and HWCO [Peng and Wang 2007] crossover operators described already for different representations also were designed with the aim of faster convergence. Faster convergence helps in applications where the solution space is large and at the same time sparse as in the K-anonymity problem discussed by Lunacek et al. [2006]. GA time need not be unnecessarily wasted on invalid search areas which may be the case with basic crossover operators. Crossover operators that are specially designed to handle such issues show a better performance when compared to their basic counterparts. Another important issue is that applications that require multi-objective optimization tend to converge slower. Such applications also require the design of special crossover operators to handle the slower convergence issues.

6.2.2. Avoiding Premature Convergence. Initially due to the longer time taken by the GAs, the termination condition stopped the process from prolonging beyond a certain number of generations. This affected the quality of the solution since reducing the number of generations means reduction in the search space. This does not guarantee the convergence to optimal solution which might be present in the part of the solution space that is not explored due to the limited number of generations. Efforts to reduce time of GA resulted in poor quality of the obtained solutions. This convergence of GAs to a suboptimal solution is called *premature convergence* [Rocha and Neves 1999].

The premature convergence problem may occur when certain genes quickly dominate the entire population which leads the GA to converge to a local optimum according to Nicoara [2009]. Since at this point all solutions become the same, further search for an optimal solution turns futile. There are many known reasons for premature convergence of GAs of which the popular ones are small population size, lack of diversity in the population [Herrera et al. 1997; Nicoara 2009; Ramadan 2013; Rocha and Neves 1999] disproportionate relationship between exploration and exploitation [Herrera et al. 1997], the genetic operators used [Ramadan 2013], less mutation rate [McGinley et al. 2011; Rocha and Neves 1999], loss of efficacy of the crossover operator [McGinley et al. 2011], incorrect application of selection pressure [McGinley et al. 2011], fitness function [Jing and Lidong 2012], and the like. The crossover operators are considered effective to solve the premature convergence problem [Herrera et al. 1997]. We will discuss the crossover operators that solve the premature convergence in the forthcoming sections.

6.2.2.1. Crossover for Maintaining Exploration and Exploitation Levels. Premature convergence can be avoided when the exploration and the exploitation levels are maintained properly. High crossover rates are responsible for exploring the search space to a great extent. However, exploitation is much less in this case which leads to missing the optimal solutions. Reducing the crossover rate to an extreme on the other hand leaves the search space largely unexplored. This in turn is responsible for missing out some potential areas of the search space. Hence, a proper balance between the exploration and the exploitation levels will lead to obtaining optimal solutions at a considerable speed thereby preventing premature convergence. The following crossover operators try to avoid the premature convergence problem by maintaining proper levels of exploration and exploitation.

MMX-BLXexploit and MMX-BLXexplore – There are three main steps in a BLX-operation namely, the definition of the size of the child's i^{th} subchromosome, modification of the attributes and inheriting features for feature selection process. These two operators maintain the exploitation and exploration [Roy et al. 2015] levels such that convergence does not occur too early (taken care by exploration) and not too late (taken care by exploitation).

Herrera et al. [2007] proposed four crossover operators based on fuzzy connectives in order to maintain the exploration and exploitation levels so as to help delay the convergence only longer enough to get a good solution and not too longer.

McGinley et al. [2011] proposes Euclidean distance based adaptive control on crossover and mutation for maintaining the exploration and exploitation levels.

Very few crossover operators fall under this category as it can be seen. There are no known direct methods that measure the exploration and exploitation [Crepinsek et al. 2013]. This introduces difficulties in designing crossover operators that maintain proper exploration and exploitation. The decision on the levels of exploration-exploitation is completely problem specific. Applications requiring uni-modal function optimization requires less exploration when compared to applications that require multi-modal function optimization. Furthermore, the levels of exploration and exploitation may be different for different applications with different number of objectives to be solved.

6.2.2.2. Crossover for Maintaining Diversity of Population. Diversity is the difference among the members of the population. Generally, it is believed that high diversity results in better performance. However, when the population is highly diverse, the convergence is slowed down. According to McPhee and Hopper [1999], "Progress in evolution depends fundamentally on the existence of variation of population. Unfortunately, a key problem in many Evolutionary Computation (EC) systems is the loss of diversity through premature convergence. This lack of diversity often leads to stagnation, as the system finds itself trapped in local optima, lacking the genetic diversity needed to escape." However, higher diversity causes time consumption which again destroys the very purpose of this group of crossover operators. Hence, proper amount of diversity should be given by the crossover operators to solve the premature convergence problem effectively. The following are the crossover operators that were specifically designed for maintaining proper diversity levels in the population, which in turn prevents the premature convergence problem.

Goldberg and Lingle's [1985] Partially Mapped Crossover Operator (PMX) maps a segment of one of the parent's chromosome to a segment of the other parent's chromosome and exchanges the remaining genes. Oliver et al. [1987] create offspring from the parents by copying the gene value and its position from the parents to the offspring based on the feasibility of the chromosome in their Cycle Crossover operator (CX). Brady's Sorted Match Crossover [Brady 1985] identifies similar subtours in the parents and generates offspring by replacing the least-cost subtour with the most-cost one. All the above-mentioned approaches aim at maintaining proper diversity in the population so as to prevent the premature convergence problem.

The chaotic crossover operator [Demirci et al. 2015] produces a single child from two parents by weighting the genes of both the parents considered using chaotic random number generator for getting near-global optimal solutions. This crossover operator improves the diversity of the population using the chaotic randomness thereby avoiding the local optima problem.

Frequency Crossover – Frequency of alleles of the genes is calculated between the parents and common alleles are inherited by the offspring in the same gene position as that of the parents and the remaining genes are filled according to the order of the genes in the parent chromosomes. This helps in stabilizing the order of the genes across generations [Ramadan 2013]. This initially reduced the diversity and backed premature convergence. Hence, mutation techniques were used along with this crossover operator to introduce diversity and reduce premature convergence.

Adaptive probabilities of mutation and crossover rates was used by Srinivas and Patnaik [1994] to control the diversity (just enough to reach optimal solution) of population thereby giving better performance.

Li et al. [2004] proposed entropy based adaptive control on diversity measure for crossover and mutation so as to avoid the premature convergence problem.

Krawiec and Lichocki geometric crossover [Pawlak et al. 2014] and the EAX [Takahashi 2008] crossover described earlier also use the diversity level to overcome the premature convergence problem. The challenge regarding the crossover operators falling under this category is again the lack of a proper measure for diversity. Generally, diversity measures are problem specific. Burke et al. [2004] in their work have identified that there is no positive correlation between the fitness function used and the diversity measures in the case of GP. In such cases, there is no point in using diversity to improve performance (fitness). Hence, such applications should be identified and other mechanisms should be involved for performance improvement in such cases. Moreover, diversity is also not much related to the exploration and exploitation levels. Diversity does not guarantee a fit population but only gives difference in the individuals of the population according to Crepinsek et al. [2013].

6.2.2.3. Crossover Repair Operators. Some repair operators were proposed for the purpose of repairing the adverse effects of crossover such that they are applied before or after the crossover and thereby help in preventing the premature convergence. The following are the crossover repair operators designed to tackle the premature convergence problem;

Determinism Operator [Durand and Alliot 1998] – The adapted crossover operator optimizes the global function by optimizing each variable separately considering the problem of completely separable functions. A single child is created from two parents, by choosing a gene that locally fits better from either parent, with an additional operator called the *determinism operator*, which helps in avoiding the premature convergence problem.

Random Offspring Generation (ROG) tests an individual's genetic material, before crossover is performed, and in case they share common genetic material, one or both the offspring are randomly generated since randomness increases diversity by introducing the GA to new and unexplored subsections of the solution space [Rocha and Neves 1999].

The cleanup operator was designed specifically to handle premature convergence in the Travelling Salesman Problem (TSP) [Mitchell et al. 2000]. This operator was used to find and correct errors (like unequal tour length, visiting a city more than once, and not visiting certain cities in the tour) in the offspring formed as the result of crossover operation. This reduces the circulation of the invalid offspring thereby promoting chances of obtaining a global optimum quickly. However, the operator was designed for a specific domain which cannot be applied to other domains.

Duplication operator essentially creates the clone of the best fit individual and the fabrication operator uses the statistical information from the elite chromosome base to fabricate a new individual after which the regular GA operations are done. These new operators help in faster convergence to a better solution according to Chang et al. [1995]. However, they also claim that the statistical information of the area of search space, which may hold the potential solutions that is not present in the population, is not taken into consideration here.

Repair operators are also problem specific and the necessity for the repair procedure should be determined for the applications before the design of the repair operator.

For a more clear understanding of the existing crossover operators classified under our classification scheme, refer to Table 1 in the appendix.

7. CROSSOVER-RELATED ISSUES

This section describes in detail the various issues related to the crossover operators.

7.1. Crossover vs. Mutation

Holland [1992] was the first one to say that crossover was better when compared to mutation, based on GA's better performance in the presence of high crossover probabilities and low mutation probabilities along with the support from the schema theorem and the building block hypothesis. However, later on, owing to the pure empirical nature of these results, the problem of comparing mutation and crossover was considered to comparing "apples and oranges". This claim proved to be the starting point of the debate on which is a better operator, crossover or mutation.

Deep and Thakur [2007] clearly state some differences between crossover and mutation and they are (1) crossover exploits search space, whereas mutation explored it, (2) crossover's effectiveness dropped with the increase in similarities between organisms in the population, whereas the necessity of mutation increased with the decrease in the variation in the population, and (3) crossover preserves (good or bad genes), whereas mutation destroys (good or bad genes). In addition, Deep and Thakur [2007] also proved that the crossover operator is superior to mutation in producing good fitness results when a longer binary string representation is used. There are many research papers that argue on this topic. Yoon and Kim [2012] present the following valuable results in their work on using four crossover operators, namely the box, extended box, line, and extended line crossover operators, in the presence and absence of Gaussian mutation and fine mutation.

—No best crossover operator was found for all types of problems.
—Extended-box crossover performed comparatively better in the absence of mutation.
—Extended-box crossover worsened the GA performance with increasing mutation rate (since it has the mutation function built in).
—All other crossover operators performed better in the presence of mutation.

We discuss the effect of crossover bias in the next section.

7.2. Crossover Bias

The crossover bias is the favoring of specific portions in the search space by crossover, i.e., crossovers have a tendency to favor the area towards the center of the search space. Crossover bias was first introduced by Eshelman et al. [1997]. However, crossover bias has been studied critically and heuristic methods were proposed to avoid real-coded crossover bias, which was incomplete theoretically [Tsutsui 1998; Tsutsui and Goldberg 2001; Someya and Yamamura 2005].

Crossover bias is defined differently, based on the representation of the chromosomes according to Yoon and Kim [2012]. The bias that favors the center of the search space is called *crossover bias* in real-coded crossovers whereas, in binary representation, the number of bits and their position in the parents inherited by the offspring are considered for bias, and in tree encoding used by GP, bias focuses on bloat (large size solutions).

In the presence of selection, bias forces the search towards favored areas. We proceed in the right direction as long as the bias is towards the optimum and GA performance becomes poor when the bias is not towards optimum. In the case of the absence of knowledge on the spatial distribution of the fitness landscape it is better not to have a bias operator. Yoon and Kim [2012] have experimentally proved that higher mutation rates decrease the effect of bias. They have also proved that, though increasing the mutation rate reduces the effect of bias, elimination of bias is impossible even by increasing the mutation rates further.

Vekaria and Clack [1999] have described four different biases for adaptive recombination. They are (1) directional bias that exists if alleles are either favored or not favored for their credibility, (2) credit bias which is the degree at which an allele gets

favored with respect to its credibility, (3) initialization bias that exists if alleles get favored during initialization without knowing their credibility, and (4) hitchhiker bias that exists if alleles get favored when they do not contribute to the fitness increase.

7.3. Crossover Rate

There is strong evidence that proves crossover rate and mutation rate are very important for the betterment of GAs [Ochoa et al. 1999; Deb and Agrawal 1999]. Initially, the apt crossover and mutation rates for a problem were found out using the trial-and-error method. However, the optimal crossover and mutation rates for different application scenarios are completely different.

Lin et al. [2003] were the first to design adaptive crossover and mutation rates in their work where they adapt the crossover and mutation rates using the fitness of the children generated in subsequent generations. Lin et al [2003] have proved experimentally that adaptive crossover and mutation rates significantly improve the performance of GAs.

Korejo et al. [2013] have proposed an adaptive crossover that modifies the crossover operator's probabilities that considers global behavior of the population for subsequent generations. They simultaneously apply more than one crossover operators in the genetic process. This way, better individuals in the search space are guaranteed.

7.4. Other Crossover-Related Issues

Lin and Yao [1997] empirically studied an EA with four recombination operators to prove their assumption that 'step size' of recombination operators plays an important role in GAs. They studied an EA without a mutation operator on several problems. However, they have arrived at an important conclusion that with the binary encoding scheme, GAs with a large search step size are generally better than GAs with a small step size for most problems.

Spears [2000], in his work, found that the crossover operators are beneficial in the presence of less local optima as a result of the study on the effects of the construction and destruction probabilities of the crossover operators.

There are many claims stating the importance of the crossover operators in reducing the run time of GAs. While Jansen and Wegener [2002, 2005] claim that the crossover operator reduces the GAs run time for the Real Royal Road problem, Lehre and Yao [2008], Doerr et al. [2008], and Doerr and Theile [2009] proved the same for the two-path and all-pairs-shortest path problems, respectively. However, Richter et al. [2008] proved that GA requires less time using only mutation when compared to GAs using mutation along with crossover operation for the Ignoble Trails problem. Yu et al. [2010] present an approach to compare the runtime of a GA with and without the crossover operators using the Markov Chain Switching Theorem. The analysis of their results show how much the crossover operators they have considered are better than the mutation being considered. In addition, Yu et al. [2010] also claim that for the interactions between GA operators like mutation and crossover, the effects of the size of population and selection pressure (the pressure caused by the selection mechanism employed – generally tends to favor best fit chromosomes) are other important issues that play a major role in determining the contribution of the crossover operators to the performance of the GAs.

8. CONCLUSIONS

We have classified the existing crossover operators based on the representation, the working principle that creates the offspring, and the contribution of crossover for improving the GA performance of quality of the solution obtained and the speed with respect to the applications for which they were designed. We have also discussed crossover-related issues. As a result of our survey on the various existing crossover

operators, we suggest the following future work in the area of designing new crossover operators:

Application specific crossover operator design can be looked into rather than using general crossover operators for various applications. Moreover, certain crossover operators are designed for representations rather than applications that might not always be productive. For example, permutation-encoding-based crossover operators can be better when designed specifically for the application rather than commonly for the encoding.

The major problem that hinders GA from obtaining the global optimum is not being able to concentrate on the fruitful areas of the search space especially when the search space is large. This can be easily handled by the intelligent design of crossover operators such that potential solution areas of the search space are identified and concentrated on to find the global optimum. For example, real encoding has this problem of very large and continuous search space. Here the potential solution areas should be identified through crossover operators such that they concentrate on a particular region of the search space that is close to the global optima rather than widening the search to fruitless areas of the search space.

Crossover generally has the tendency of circulating the same sets of genes. However, the use of different crossover operators for the same set of initial population either adaptively or normally varies the alleles for genes. Compound and hybrid crossover operators help in combining the effects of more than one crossover operators which can be used wisely to create even better crossover operators so as to obtain better offspring.

Heuristic design of crossover generally gives promising results since the requirement (whether it is obtaining the optimal solution, or obtaining faster convergence, or avoiding the local optima problem, or so on) can be aimed at precisely.

Crossover is capable of improving the quality of solutions obtained using GA. There are many existing crossover operators that help in improving the quality of solutions obtained using GA. However, reaching the global optimum is still a problem for some applications. Such applications can be identified and application-specific crossover operators to improve the GA performance can be designed.

Adaptive crossover is a class where future improvements are vast. This is because the prediction of a better crossover rate or a better crossover operator for the results based on factors like diversity of the population, proximity of the solution to global optimum, and the like, in the previous generation is very important for certain applications. However, finding the right amount of adaptivity is a great challenge in designing adaptive crossover operators.

Multiple parent crossovers can be improved effectively by finding out the better alleles for the genes based on heuristics. This in fact gives the double advantage of quality improvement as well as faster convergence to GA. Combining multiple parent crossovers with effective search heuristics to concentrate on potential areas of the search space can improve both the quality of the GA solution and its speed.

Multiple offspring method can be used to narrow the search towards global optimum and then some other crossover operator that concentrates on these promising areas can be done (adaptive) to get the optimal solution.

The role of crossover in preventing the premature convergence is studied extensively and is one of the promising areas of research in GA. Obtaining good quality solutions and speedy convergence is very important in GAs. Future work in designing crossover operators should aim at understanding the relationship between obtaining quality solutions and speed of GA and wisely deploy it in the design of crossover operators.

Crossovers that can introduce diversity to the population can be developed without the need for mutation as a separate step, thus reducing a level of time consumption without compromising on the goodness of the solution.

As we already saw, controlling the levels of exploitation and exploration is of huge interest for improving the quality of solutions and speed of GA such that getting trapped in local optimum is avoided. However, control of these levels needs direct methods that measure exploitation and exploration, which are currently not available. Thus, designing new measures that capture the levels of exploration and exploitation aptly and using this information for the design of new crossover operators can be examined as a potential area of research as maintaining the exploration and the exploitation levels improve both the quality of solutions and speed of GA.

Problem-specific diversity measures should be devised to measure the diversity of solutions and they should be used in the crossover design such that the quality of solutions and speed of GA is improved. Furthermore, as we have already seen, using diversity to improve fitness under cases where there is no positive correlation may prove waste. Hence, such applications should be identified so that other mechanisms can be involved for performance improvement.

"Mutation and crossover operators that influence exploration and exploitation mostly depend on representation of individuals", according to Crepinsek et al.yy [2013]. As we have already seen, representation is a very important factor of GA. We should understand the strong relationship between the representations, the quality of solutions, and the speed of GA as each of the three have strong effects on the other two. What is even more interesting is the special bond that crossover shares with each of these aspects of GA. We have seen that crossover operators play a pivotal role in the improvement of all the three cases. Future design of the crossover operators should take into account the most important dimension, i.e., the application, along with these three dimensions. A future crossover design can consider all four dimensions that we have mentioned above. This will add complexity to the design of crossover. However, if such crossover operators are designed, GAs will undisputedly be the leading optimization algorithms.

REFERENCES

Tisha Agarwal and Kanchan Singh. 2013. Using new variation crossover operator of genetic algorithm for solving the Traveling Salesmen Problem. *MIT International Journal of Computer Science and Information Technology* 3, 1, (Jan. 2013), 35–37.

Jesus S. Aguilar-Ruiz, Raul Giraldez, and Jose C. Riquelme. 2007. Natural encoding for evolutionary supervised learning. *IEEE Transactions on Evolutionary Computation* 11, 4 (Aug. 2007), 466–479.

Plamen Angelov. 2001. Supplementary crossover operator for genetic algorithms based on the center-of-gravity paradigm. *Control and Cybernetics* 30, 2 (2001), 159–176.

Michael Baake. 2008. *Repeat Distributions from Unequal Crossovers*. Banach Center Publications, Institute of Mathematics, Polish Academy of Sciences Warszawa, 80, (Mar. 2008), 53–70.

Amit Banerjee and Rajesh N. Dave. 2010. Context sensitive crossover operator for clustering applications. In *Proceedings of the IEEE Conference on Evolutionary Computation (CEC'10).* IEEE, 1–8.

Clare Bates Congdon. 2002. Gaphyl: An evolutionary algorithms approach for the study of natural evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02).* IEEE, New York, 1057–1064.

John E. Beasley and P. C. Chu. 1996. A genetic algorithm for the set covering problem. *European Journal of Operational Research* 94, 2 (Oct. 1996), 392–404.

Josh C. Bongard. 2010. A probabilistic functional crossover operator for genetic programming. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'10).* IEEE. 925–932.

R. M. Brady. 1985. *Optimization* Strategies gleaned from biological evolution. *Nature* 317, 804–806.

E. Burke, S. Gustafson, and G. Kendall. 2004. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation* 8, 1, 47–62.

G. Chakraborty and K. Hoshi. 1999. Rank based crossover – A new crossover technique to improve convergence in genetic algorithms. In *Proceedings of the Congress on Evolutionary Computation (CEC'99).* IEEE, 1595–1602.

K. Y. Chan, M. E. Aydin, and T. C. Fogarty. 2003. A Taguchi method-based crossover operator for the parametrical problems. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'03)*. 971–977.

P. C. Chang, Y. W. Wang, and C. H. Liu. 1995. New operators for faster convergence and better solution quality in modified genetic algorithm. In *Advances in Natural Computation*. Springer, 983–991.

Janet Clegg, James Alfred Walker, and Julian Francis Miller. 2007. A new crossover technique for cartesian genetic programming. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'07)*. IEEE, 1580–1587.

M. Coli, G. Gennuso, and P. Palazzari. 1996. A new crossover operator for genetic algorithms. In *Proceedings of the International Conference on Evolutionary Computation (CEC'96)*. IEEE, 201–206.

Matej Crepinsek, Shih-Hsi Liu, and Usa Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys* 45, 3. DOI:10.1145/2480741.2480752.

Ioannis G. Damousis, Anastasios G. Bakirtzis, and Petros S. Dokopoulos. 2004. A solution to the unit-commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power Systems* 19, 2 (May. 2004), 1165–1172.

Lawrence Davis. 1989. Adapting operator probabilities in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. David J. Schaffer, (Ed.) Morgan-Kaufmann Publishers, San Francisco, CA, 61–69.

Kalyanmoy Deb. 2000. Introduction to representations. In *Evolutionary Computation 1: Basic Algorithms and Operators*. Thomas Baeck, D. B.Fogel and Z. Michalewicz (Eds.). Institute of Physics Publishing, Bristol, 127–131.

Kalyanmoy Deb and R. B. Agrawal. 1995. Simulated binary crossover for continuous search space. *Complex Systems* 9, 115–148.

Kalyanmoy Deb and Samir Agrawal. 1999. Understanding interactions among genetic algorithm parameters. In *Foundations of Genetic Algorithms V*. 265–286.

Kalyanmoy Deb and Hans-Georg Beyer. 1999. *Self-Adaptive Genetic Algorithms with Simulated Binary Crossover*. Tech. Report No.CI-61/Department of Computer Science/XI, University of Dortmund, 44221 Dortmund, Germany.

Kusum Deep and Hadush Mebrahtu. 2011. New variations of order crossover for travelling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics* 2, 1, 2–13.

Kusum Deep and Manoj Thakur. 2007. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation* 193, 1 (Oct. 2007), 211–230.

H. Demirci, A. T. Ozcerit, H. Ekiz, and A. Kutlu. 2015. Chaotic crossover operator on genetic algorithm. In *Proceedings of 2nd International Conference on Information Technology*.

Benjamin Doerr and Madeleine Theile. 2009. Improved analysis methods for crossover-based algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'09)*. 247–254.

Benjamin Doerr, Edda Happ, and Christian Klein. 2008. Crossover can provably be useful in evolutionary computation. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'08)*. 539–546.

N. Durand and J. M. Alliot. 1998. Genetic crossover operator for partially separable functions. In *Proceedings of 3rd Annual Conference on Genetic Programming*. Morgan-Kaufmann. 487–494

A. E. Eiben, P. E. Raue, and Z. S. Ruttkay. 1994. Genetic algorithms with multi-parent recombination. In *Proceedings of the Parallel Problem Solving from Nature III (PPSN'94)*. 78–87.

Larry J. Eshelman, Keith E. Mathias, and J. David Schaffer. 1997. Crossover operator biases: Exploiting the population distribution. In *Proceedings of the International Conference on Genetic Algorithms*. 354–361.

Larry J. Eshelman, Richard A. Caruana, and J. David Schaffer. 1989. Biases in the crossover landscape. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufman, 10–19.

B. R. Fox and M. McMahon. 1990. Genetic operators for sequencing problems. In *Proceedings of the Workshop on the Foundations of Genetic Algorithms* 284–300.

Adrien Goeffon, Jean-Michel Richer, and Jin-Kao Hao. 2006. A distance-based information preservation tree crossover for the maximum parsimony problem. In *Proceedings of the Parallel Problem Solving from Nature (PPSN'06)*. 761–770.

David E. Goldberg and Robert Lingle Jr. 1985. Alleles, loci, and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*. 154–159.

Thomas Grueninger and David Wallace. 1996. *Multi-modal Optimization Using Genetic Algorithms*. Technical Report 96.02, CAD Lab, Massachusetts Institute of Technology, Cambridge, MA.

Mark Harman, Robert Hierons, and Mark Procter. 2002. A new representation and crossover operator for search-based optimization of software modularization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)*. 1351–1358.

F. Herrera, M. Lozano, and A. M. Sanchez. 2003. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* 18, 3 (Mar. 2003), 309–338.

F. Herrera, M. Lozano, and A. M. Sanchez. 2005. Hybrid crossover operators for real-coded genetic algorithms: An experimental study. *Soft Computing* 9, 4 (Apr. 2005), 280–298. DOI:10.1007/s00500-004-0380-9.

F. Herrera, M. Lozano, and J. L. Verdegay. 1996. Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms. *International Journal of Intelligent Systems* 11, 12 (Dec. 1996), 1013–1041.

F. Herrera, M. Lozano, and J. L. Verdegay. 1997. Fuzzy connectives based crossover operators to model genetic algorithms population diversity. *Fuzzy Set Systems* 92, 1 (Nov. 1997), 21–30.

F. Herrera, M. Lozano, and J. L. Verdegay. 1998. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review* 12, 4 (Aug. 1998), 265–319.

M. I. Heywood and A. N. Zincir-Heywood. 2002. Dynamic page-based linear genetic programming. *IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics* 32, 3, 380–388.

Shinn-Ying Ho, Chia-Cheng Liu, and Soundy Liu. 2002. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters* 23, 1495–1503.

John H. Holland. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Cambridge, MA.

David Jackson. 2004. *Modified Crossover Operators for Protein Folding Simulation with Genetic Algorithms.* Master's Thesis, Carleton University, Ottawa.

Thomas Jansen and Ingo Wegener. 2002. The analysis of evolutionary algorithms - A proof that crossover really can help. *Algorithmica* 34, 1 (Sep. 2002), 47–66.

Thomas Jansen and Ingo Wegener. 2005. I. Real royal road functions - Where crossover provably is essential. *Discrete Applied Mathematics* 149, 1–3 (Aug. 2005), 111–125.

Da-Jiang Jin and Ji-Ye Zhang. 2007. A new crossover operator for improving ability of global searching. In *Proceedings of the 6th International Conference on Machine Learning and Cybernetics*. 2328–2332.

J. Jing and M. Lidong. 2012. The strategy of improving convergence of genetic algorithm. *Telkomnika* 10, 8, 2063–2068.

Ali Karc. 2004. Imitation of bee reproduction as a crossover operator in genetic algorithms. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence (PRICAI'04).* 1015–1016.

Yılmaz Kaya, Murat Uyar, and Ramazan Tekdn. 2011. *A Novel Crossover Operator for Genetic Algorithms: Ring Crossover*. CoRR abs/1105.0355.

A. Korejo, Z. U. A. Khuhro, F. A. Jokhio, N. Channa, and H. A. Nizamani. 2013.An adaptive crossover operator for genetic algorithms to solve the optimization problems. *Sindh University Research Journal (Science Series)* 45, 2, 333–340.

John. R. Koza. 1992. *Genetic Programming*. MIT Press, Cambridge, MA.

Rakesh Kumar and Jyotishree. 2012. Novel encoding scheme in genetic algorithms for better fitness. *International Journal of Engineering and Advanced Technology* 1, 6 (Aug. 2012), 214–218.

Rakesh Kumar, Girdhar Gopal, and Rajesh Kumar. 2013. Novel crossover operator for genetic algorithm for permutation problems. *International Journal of Soft Computing and Engineering* 3, 2 (May 2013), 252–258.

George S. Ladkany and Mohamed B. Trabia. 2012. A genetic algorithm with weighted average normally-distributed arithmetic crossover and twinkling. *Applied Mathematics* 3, 1220–1235. http://dx.doi.org/10.4236/am.2012.330178

W. Langdon and R. Poli. 2002. *Foundations of Genetic Programming*. Springer-Verlag.

Rhydian Lewis and Ben Paechter. 2004. New crossover operators for timetabling with evolutionary algorithms. In *Proceedings of 5th International Conference on Recent Advances in Soft Computing (RASC'04).* 189–195.

Per Kristian Lehre and Xin Yao. 2008. Crossover can be constructive when computing unique input output sequences. In *Proceedings of Simulated Evolution and Learning (SEAL'08)*. 595–604.

P. O. Lewis. 1998. A genetic algorithm form maximum-likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology Evolution* 15, 3 (Mar. 1998), 277–283.

Jingzhi Li, Lishan Kang, and Zhijian Wu. 2003. An adaptive neighborhood-based multi-parent crossover operator for real-coded genetic algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*. 14–21.

Fan Li, Qi-He Liu, Fan Min, and Guo-Wei Yang. 2005. A new crossover operator based on the rough set theory for genetic algorithms. In *Proceedings of the 4th International Conference on Machine Learning and Cybernetics.* 2907–2912.

M. Li, Z Cai, and G. Sun. 2004. An adaptive genetic algorithm with diversity-guided mutation and its global convergence property. *Journal of Central South University of Technology* 11, 3, 323–327.

Gunar E. Liepins and Micheal D. Vose. 1992. Characterizing crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence* 5, 1, 27–34.

Guangming Lin and Xin Yao. 1997. Analysing crossover operators by search step size. In *Proceedings of the IEEE International Conference on Evolutionary Computation.* 107–110.

Wen-yang Lin, Wen-yung Lee and Tzung-pei Hong. 2003. Adapting crossover and mutation rates in genetic algorithms. *Journal of Information Science and Engineering* 19, 889–903.

M. Lunacek, D. Whitley, and I. Ray. A crossover operator for the *k*-anonymity problem. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'06).* ACM,1713–1720.

H. Matsuda. 1996. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. In *Proceedings of the Pacific Symposium on Biocomputing.* 512–523.

B. McGinley, J. Maher, C.O'Riordan, and F. Morgan. 2011. Maintaining healthy population diversity using adaptive crossover, mutation and selection (ACROMUSE). *IEEE Transactions on Evolutionary Computation* 15, 5, 692–714.

N. F. McPhee and N. J. Hopper. 1999. Analysis of genetic diversity through population history. In *Proceedings of the 1st Genetic and Evolutionary Computation Conference*. 1112–1120.

Z. Michalewicz, G. Nazhiyath, and M. Michalewicz. 1996. A note on usefulness of geometrical crossover for numerical optimization problems. In *Proceedings of the 5th Annual Conference on Evolutionary Programming*. MIT Press, Cambridge, MA, 305–312.

Z. Michalewicz. 1994. *Genetic Algorithms+Data Structures = Evolution Programs* (2nd ed.). Springer-Verlag, New York.

Julian F. Miller and Peter Thomson. 2000. Cartesian genetic programming. In *Proceedings of the 3rd European Conference on Genetic Programming.* 121–132.

Alfonsas Misevicius and Bronislovas Kilda. 2005. Comparison of crossover operators for the quadratic assignment problem. *Information Technology and Control* 34, 2, 109–119.

G. G. Mitchell, D. O'Donoghue, and A. Trenaman. 2000. A new operator for efficient evolutionary solutions to the travelling salesman problem. In *Proceedings of Applied Informatics*.771–774.

M. Mitchell. 1999. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, MA.

Alberto Moraglio and Riccardo Poli. 2004. Topological interpretation of crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'04).* 1377–1388.

Alberto Moraglio. 2007. *Towards a Geometric Unification of Evolutionary Algorithms*. PhD dissertation, Department of Computer Science, University of Essex, 392.

Sergio Morales-Ortigosa, Albert Orriols-Puig, and Ester Bernad´o-Mansilla. 2008. New crossover operator for evolutionary rule discovery in XCS. In *Proceedings of the 8th International Conference on Hybrid Intelligent Systems (HIS'08).* 867–872.

Devasenathipathi N. Mudaliar and Nilesh K. Modi. 2013. Unraveling travelling salesman problem by genetic algorithm using M-crossover operator. In *Proceedings of the International Conference on Signal Processing Image Processing and Pattern Recognition (ICSIPR'13*). 127–130.

Christine L. Mumford. 2006. New order-based crossovers for the graph coloring problem. In *Parallel Problem Solving from Nature* –IX *(PPSN'06).* 880–889.

H. Nazif and L. S. Lee. 2010. Optimized crossover genetic algorithm for vehicle routing problem with time windows. *American Journal of Applied Sciences* 7, 1, 95–101.

E. S. Nicoara. 2009. Mechanisms to avoid the premature convergence of genetic algorithms. *University of Ploiesti Bulletin, Math*. 61, 1, 87–96.

Gabriela Ochoa, Inman Harvey, and Hilary Buxton. 1999. On recombination and optimal mutation rates. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'99).* 488–495.

I. Oliver, D. Smith, and J. Holland. 1987. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the 2nd International Conference on Genetic Algorithms.* 224–230.

Isao Ono, Hajime Kita, and Shigenobu Kobayashi. 2003. *A Real-Coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover.* Advances in Evolutionary Computing, Natural Computing Series (2003), 213–237.

Viktor Otevrel and Zbynek Raida. 2007. Mean-adaptive real-coding genetic algorithm and its applications to electromagnetic optimization (Part One). *Radio Engineering* 16, 3 (Sep. 2007), 19–29.

Abdoun Otman and Abouchabaka Jaafar. 2011. A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *International Journal of Computer Applications* 31, 11 (Oct. 2011), 49–57.

Nikhil R. Pal, Malay K. Kundu, and Suchismita Nandi. 1998. Application of a new genetic operator in feature selection problems. In *Proceedings of the IEEE Region 10th International Conference on Global Connectivity in Energy, Computer, Communication and Control.* 37–40.

Karoly F. Pal. 1994. Selection schemes with spatial isolation for genetic optimization. In *Proceedings of the Parallel Problem Solving from Nature (PPSN'94)*. 170–179.

R. Patel and M. M. Raghuwanshi. 2011. Multi-objective optimization using multi parent crossover operators. *Journal of Emerging Trends in Computing and Information Sciences* 2, 2, 99–105.

G. Pavai and T. V. Geetha. 2015. New real coded crossover operators for genetic algorithms based on incomplete dominance and gene memory. *Advances in Natural and Applied Sciences* 9, 6, 568–573.

Tomasz P. Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. 2014. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines* 16, 3, 351–386. DOI:10.1007/s10710-014-9239-8

Li Peng and Wei Wang. 2007. Adaptive genetic algorithm with heuristic weighted crossover operator based hysteresis identification and compensation. In *Proceedings of IEEE International Conference on Control and Automation*. 769–773.

Stjepan Picek, Marin Golub, and Domagoj Jakobovic. 2011. Evaluation of crossover operator performance in genetic algorithms with binary representation. In *Proceedings of the 7th International Conference on Intelligent Computing: Bio-Inspired Computing and Applications (ICIC'11)*. 223–230.

Riccardo Poli and Nicholas F. McPhee. 2000. *Exact GP Schema Theory for Headless Chicken Crossover and Subtree Mutation.* Technical Report: CSRP-00-23 December 2000.

Nicholas J. Radcliffe. 1992. *Non-Linear Genetic Representations*. In *Proceedings of Parallel Problem Solving from Nature*, R. Manner, and B. Manderick, (Ed.) Elsevier, 259–268.

Rosshairy Abd Rahman and Razamin Ramli. 2013. Average concept of crossover operator in real coded genetic algorithm. *International Proceedings of Economics Development and Research* 63, 15 (May 2013), 73–77.

S. Z. Ramadan. 2013. Reducing premature convergence problem in genetic algorithm: application on travel salesman problem. *Journal of Computing and Information Sciences* 6, 1, 410–418.

T. H. Reijmers, R. Wehrens, F. D. Daeyaert, P. J. Lewi, and L. M. C. Buydens. 1999. Using genetic algorithms for the construction of phylogenetic trees: Application to G-protein coupled receptor sequences. *Biosystems* 49, 1 (Jan. 1999), 31–43.

J. Neal Richter, Alden Wright, and John Paxton. 2008. Ignoble trails - Where crossover is provably harmful. In *Proceedings of Parallel Problem Solving from Nature (PPSN'08)*. 92–101.

M. Rocha and J. Neves. 1999. Preventing premature convergence to local optima in genetic algorithms via random offspring generation. In *Proceedings of 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE'99)*. Springer, 127–136.

A. Roy, J. D. Schaffer, and C. B. Laramee 2015. New crossover operators for multiple subset selection tasks. *Computing Communication and Collaboration* 3, 1, 42–62.

M. Sadeghi, R. Sadeghi, S. A. Mousavi, and S. Khanmohammadi. 2013. Improved genetic algorithms by means of fuzzy crossover operators for revenue management in airlines. W*orld Applied Sciences Journal* 2, 6, 838–846.

B. Saravanan, Siddharth Das, Surbhi Sikri, and D. P. Kothari. 2013. A solution to the unit commitment problem—a review. *Frontiers in Energy* 7, 2, 223–236.

Kumara Sastry and David E. Goldberg. 2002. Analysis of mixing in genetic algorithms: A survey. IlliGAL Report No. 2002012, University of Illinois at Urbana-Champaign, Urbana, IL.

D. Schlierkamp-Voosen. 1994. Strategy adaptation by competition. In *Proceedings of the 2nd European Conference on Intelligent Techniques and Soft Computing*. 1270–1274.

R. R. Sharapov. 2007. *Genetic Algorithms: Basic ideas, Variants and Analysis, Vision Systems: Segmentation and Pattern Recognition.* G. Obinata and A. Dutta (Ed.), 407–422.

S. K. Sharma and G. W. Irwing. 2003. Fuzzy coding of genetic algorithms. *IEEE Transactions on Evolutionary Computation* 7, 4 (Aug. 2003), 344–355.

Hiroshi Someya and Masayuki Yamamura. 2005. A robust real-coded evolutionary algorithm with toroidal search space conversion. *Soft Computing* 9, 4.254–269.

William M. Spears. 2000. *Evolutionary Algorithms: The Role of Mutation and Recombination.* Springer-Verlag, Berlin.

M. Srinivas and L. M. Patnaik. 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 24, 656–667.

Hal Stringer and Annie S. Wu. 2005. *Variable-Length Genetic Algorithms and an Analysis of Changes in Chromosome Length Absent Selection Pressure.* School of Computer Science, University of Central Florida, Orlando.. http://www.cs.ucf.edu/~stringer/ECJ%t20Article.pdf.

Fanchen Su, Fuxi Zhu, Zhiyi Yin, Haitao Yao, Qingping Wang, and Wenyong Dong. 2009. New crossover operator of genetic algorithms for the TSP. In *Proceedings of IEEE International Joint Conference on Computational Sciences and Optimization*. 666–669.

Stephen Swift, Allan Tucker, Jason Crampton, and David Garway-Heath. 2007. An improved restricted growth function genetic algorithm for the consensus clustering of retinal nerve fibre data. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'07)*. 2174–2181.

G. Syswerda. 1989. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufman, 2–9.

Ryouei Takahashi. 2008. A methodology of extended changing crossover operators to solve the traveling salesman problem. In *Proceedings of the 4th International Conference on Natural Computation (ICNC'08)*. 263–269.

Shigeyoshi Tsutsui and Ashish Ghosh. 1998. A study on the effect of multi-parent recombination in real coded genetic algorithms. In *Proceedings of the IEEE International Conference on Evolutionary Computation*. 828–833.

Shigeyoshi Tsutsui and David E. Goldberg. 2001. Search space boundary extension method in real-coded genetic algorithms. *Information Sciences* 133, 3-4, 229–247.

Shigeyoshi Tsutsui. 1998. Multi-parent recombination in genetic algorithms with search space boundary extension by mirroring. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature V (PPSN'98)*, Amsterdam, The Netherlands, 428–437.

Gokturk Ucoluk. 2002. Genetic algorithm solution of the TSP avoiding special crossover and mutation. *Intelligent Automation and Soft Computing* 8, 3, 265–272.

Kanta Vekaria and Chris Clack. 1999. Biases introduced by adaptive recombination operators. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'99)*. 670–677.

Dana Vrajitoru. 1998. Crossover improvement for the genetic algorithm in information retrieval. *Information Processing and Management* 34, 4 (Jul. 1998), 405–415.

Thomas Weise. 2009. *Global Optimization Algorithms Theory and Application*. Retrieved from www.it-weise.de/.

Darrell Whitley, Doug Hains, and Adele Howe. 2010. A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover. In *Parallel Problem Solving from Nature (PPSN'10) XI*. 566–575.

Darrell Whitley, Timothy Starkweather, and Daniel Shaner. 1991. *Travelling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination*. Handbook of Genetic Algorithms, L. Davis (Ed.) Van Nostrand Reinhold, 350–372.

A. Wright. 1991. Genetic algorithms for real parameter optimization. Rawlin GJE, ed. In *Foundations of Genetic Algorithms 1*, Rawlin GJE, ed. Morgan Kaufmann, 205–218.

S. Yang. 2002. Adaptive non-uniform crossover based on statistics for genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)*. 650–657.

Yourim Yoon and Yong-Hyuk Kim. 2012. *The Roles of Crossover and Mutation in Real-Coded Genetic Algorithms, Bio-Inspired Computational Algorithms and Their Applications*. Shangce Gao (Ed.). Retrieved from http://www.intechopen.com/books/bio-inspired-computational-algorithms-and-their-applications/the-roles-of-crossover-and-mutation-in-real-coded-genetic-algorithms.

Masaya Yoshikawa, Kentaro Otsuka, and Hidekazu Terai. 2008. Crossover operation engine considering character inheritance. In *Recent Advances in Systems, Communications and Computers, Selected Papers from the WSEAS Conferences*. 117–122.

Yang Yu, Chao Qian, and Zhi-Hua Zhou. 2010. Towards analyzing recombination operators in evolutionary search. In *Parallel Problem Solving from Nature (PPSN'10)*. 144–153.

Qing-ke Yuan, Shi-jie Li, Ling-lian Jiang, and Wen-yan Tang. 2009. A mixed-coding genetic algorithm and its application on gear reducer optimization. *Fuzzy Information and Engineering*. *2. Advances in Intelligent and Soft Computing* 62, (2009), 753–759.

Received August 2015; revised July 2016; accepted October 2016