Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2012, Article ID 703601, 23 pages
doi:10.1155/2012/703601

Research Article

A Fuzzy Genetic Algorithm Based on Binary Encoding for Solving Multidimensional Knapsack Problems

M. Jalali Varnamkhasti[1] and L. S. Lee[2, 3]

[1] *Department of Basic Science, Islamic Azad University, Dolatabad Branch, Esfahan 84318–11111, Iran*
[2] *Laboratory of Computational Statistics and Operations Research, Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia*
[3] *Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia*

Correspondence should be addressed to M. Jalali Varnamkhasti, m.jalali_upm@yahoo.com

Received 3 December 2011; Accepted 14 February 2012

Academic Editor: Hector Pomares

The fundamental problem in genetic algorithms is premature convergence, and it is strongly related to the loss of genetic diversity of the population. This study aims at proposing some techniques to tackle the premature convergence by controlling the population diversity. Firstly, a sexual selection mechanism which utilizes the mate chromosome during selection is used. The second technique focuses on controlling the genetic parameters by applying the fuzzy logic controller. Computational experiments are conducted on the proposed techniques and the results are compared with other genetic operators, heuristics, and local search algorithms commonly used for solving multidimensional 0/1 knapsack problems published in the literature.

1. Introduction

Combinatorial optimization problems contain many decision variables, and hence exploring the search space and finding the optimal solution remain a major challenge for this type of problem. Problems, such as loading, scheduling, inventory control, and routing, occur in management fields and in many engineering design processes. Most of these problems are members of the class of NP-hard problems as classified by Garey and Johnson [1]. They are usually difficult to solve and cannot be solved in polynomial time.

The genetic algorithm (GA) [2] is a heuristic search technique that mimics natural evolution processes such as selection, crossover, and mutation operations. The selection pressure

drives the population toward better solutions while crossover, uses genes of selected parents to produce offspring that will form the next generation. Mutation is used to avoid premature convergence and consequently escapes from the local optimal. GAs have been very successful in handling hard combinatorial optimization problems.

This study aims at proposing some techniques to tackle the premature convergence of GAs by controlling the population diversity. Firstly, a new sexual selection mechanism which utilizes the mate chromosome during selection is proposed. In a classical GA, chromosomes reproduce asexually: any two chromosomes may be the parents in crossover. Gender division and sexual selection here inspire a model of gendered GA in which crossover takes place only between chromosomes of opposite sex. In this paper, the population is divided into groups of males and females. During the sexual selection, the female chromosome is selected by using the tournament selection while the male chromosome is selected based on the Hamming distance (HD) from the selected female chromosome, fitness value (FV) or the active genes (AGs) of the male chromosome.

The performance of a GA is strongly related to the equilibrium between the exploration and the exploitation of the genetic operators. Establishing a suitable relationship between the exploitation and the exploration during the GA implementation is critical for preventing the premature convergence. In this paper, fuzzy tools are used to dynamically control the parameters in the GA in order to create an appropriate balance between exploitation and exploration. In order to dynamically control the parameters in a GA, population diversity based on the phenotype and the genotype properties is applied as inputs in a fuzzy logic controller (FLC). Based on these inputs, the fuzzy selection of crossover operator, crossover probability, mutation operator, and mutation probability are produced as the outputs in the FLC. The FLC is used in four stages:

(i) fuzzy selection for crossover operator,

(ii) fuzzy selection for crossover probability,

(iii) fuzzy selection for mutation operator,

(iv) fuzzy selection for mutation probability.

The remainder of this paper is organized as follows. In Section 2, a comprehensive literature review is given. Section 3 explains in detail the proposed algorithm. Computational experiments based on the benchmark problem instances from the literature are presented in Section 4. Finally, a brief conclusion is given in Section 5.

2. Literature Review

The GAs have grown to became a significant problem-solving technique in a variety of areas which encounter problems of design optimization. It is almost impossible for one to find a uniformly optimum algorithm to resolve all problems of optimization. This is in line with the "no free lunch" theory which argues that any enhanced performance of an algorithm over a particular problem class is paid for in its performance over another exactly. Control parameters and selective operators of the evolutionary algorithms are becoming more popular because of their capabilities in solving different real world problems.

Fuzzy logic (FL) and GA can be merged in two possible cases: (i) the application of GA for solving optimization problems related with fuzzy systems and (ii) using fuzzy tools and

FL-based techniques for modelling and adapting different GA components. The GA resulting from the merging of the second case is called fuzzy genetic algorithms (FGAs) [3].

Some researchers utilized FL to dynamically control the parameters in GA in order to create an appropriate balance between exploitation and exploration. A management technique of crossover and mutation probabilities on the basis of FL was proposed by Song et al. [4] In this technique, the alteration of the average fitness value between two sequential generations is considered as the input variables. The method used by [4] encountered some problems in the fuzzy inference rules that Yun and Gen [5] later overcame by introducing a modification and a scaling factor for the normalization of the input variables.

Subbu et al. [6] presented a contrast between the performance of an FGA and a parameter-tuned GA for a nimble production application whereby parameters of the GA were dynamically scheduled by FLCs. The tuned genetic algorithm (TGA) performed superior searches but required more time for searching than the FGA. Therefore, the TGA is slower than the FGA as it uses a canonical static parameter set.

Li et al. [7] used fuzzy tools in a GA for controlling the crossover and mutation probabilities. Their FGA utilized the information of both the whole generation and the particular chromosomes. Wang et al. [8] and Wang [9] designed an FGA based on population diversity measurements whereby the fuzzy controller was employed in controlling the crossover and mutation probabilities. Liu et al. [10] designed a hybrid consisting of a fuzzy system and GA which takes into consideration the average fitness value and the best fitness value of chromosomes in each generation to allow a dynamic control of the crossover and mutation probabilities. Herrera et al. [11] and Herrera and Lozano [3, 12, 13] attempted to lay the foundations for FGAs. They formulated different possible definitions for these algorithms. They suggested an exhaustive analysis of fuzzy-adaptive GA and an adaptive approach for controlling the mutation probability based on a utilization of FLCs.

Lee and Takagi [14] proposed a method for controlling the crossover and mutation rate. They also presented an automatic fuzzy design technique which was based on a GA. Im and Lee [15] presented adaptive crossover, mutation, and selection using a fuzzy system for GA. In their study, it is not clear how a fuzzy system was based and which parameter of the GA was used for selection rules and membership functions. Jalali Varnamkhasti and Lee [16] proposed a new technique for choosing the female chromosome during sexual selection in a GA. A bilinear allocation lifetime approach is used to label the chromosomes based on their fitness value which will then be used to characterize the diversity of the population. The application of this technique is given by Jafari et al. [17] for committee neural networks.

3. Fuzzy Genetic Algorithm

The performance of an FGA depends on the design of various features, the most important of which are population, selection mechanism, crossover, and mutation operators and their probabilities and replacement strategy. In the remaining section, we discussed some of the main components of the FGA as shown in Algorithm 1.

3.1. Encoding

Chromosome encoding can be achieved by a variety of encoding methods and the choice of method is usually dependent on the optimization problem structure and precision requisites.

```
begin
    Initialise Population;
    Fitness Evaluation;
    repeat
        Population Diversity;
        Sexual Selection;
        Fuzzy Controller;
            Fuzzy Crossover;
            Fuzzy Mutation;
        Replacement;
    until the end condition is satisfied;
    return the fittest solution found;
end
```

Algorithm 1: FGA

However, the most common methods of encoding chromosomes are binary, integer, and real number encoding. The most commonly employed method of the three is binary encoding, primarily because the first works using GA employed this type of encoding. Binary encoding handles each chromosome as cord of bits, either 1 or 0. In this paper, we used standard binary encoding.

3.2. Population Diversity

In this study, the genotype and phenotype properties are both considered for the measurement of the population diversity. In the case of the phenotypic measures, the methods presented by Srinivas and Patnaik [18] and Zhu and Liu [19] are used with slight modifications [20]:

$$T_{1,t} = \frac{n_{f_i}}{N}, \qquad T_{2,t} = \frac{f_{\max,t} - f_{\text{avg},t}}{f_{\max,t}}, \tag{3.1}$$

where N: population size, n_{f_i}: number of unique fitness values in the population, $f_{\max,t}$: maximum fitness value in generation t, and, $f_{\text{avg},t}$: average fitness value in generation t.

In the case when the genotypic measure is used, an equation similar to part of the technique proposed by Jassadapakorn and Chongstitvatana [21] is used:

$$T_{3,t} = \frac{\text{HD}\left(C_{f_{\max,t}}, C_{f_{\min,t}}\right)}{L}, \tag{3.2}$$

where $\text{HD}(C_{f_{\max,t}}, C_{f_{\min,t}})$: HD between the chromosomes with the highest and the lowest FV, and L: length of the chromosome.

In this technique, the HD is calculated in each generation for only two chromosomes: the chromosome with the maximum fitness value ($C_{f_{\max}}$) and the one with the minimum fitness value ($C_{f_{\min}}$) [20].

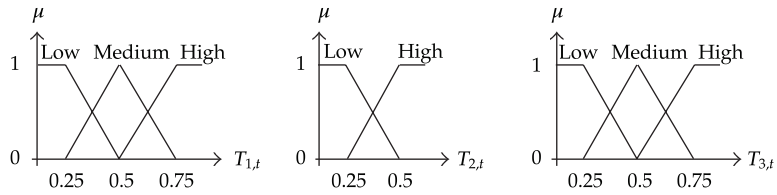

Figure 1: The set of linguistic labels associated with $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$ (source: [20]).

As regards to $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$, three membership functions are defined. The set of linguistic labels associated with $T_{1,t}$ and $T_{3,t}$ are low, medium, and high and those related to $T_{2,t}$ are low and high. The semantic (meaning) of these labels are illustrated in Figure 1 [20].

3.3. Sexual Selection

The selection mechanism drives the population toward better solutions while the crossover uses genes of the selected parents to produce offspring that will form the next generation. This operator chooses chromosomes in the present generation for use in construction of the following one. In GAs, this operator is quite similar to the biological process of natural selection. The more fitting chromosomes are better able to survive and breed. The rate of the GA convergence depends to a great extent on the selection pressure; strong pressure favours fit chromosomes and leads to fast convergence, and vice versa. Though, when the selection pressure is very high, few highly fitting chromosomes will dominate the population thus lowering the diversity due to exploring the various areas of the solution space and consequently may lead to the premature convergence. On the other hand, when the selection pressure is very weak, the GA will need a higher number of generations to find the optimum solution. A balance between the two is needed to conserve the diversity of the population and escape premature convergence.

In this paper, we use the sexual selection proposed in [20]. This technique utilizes the mate chromosome during selection. Note that the layout of the male and female chromosomes in each generation is different. During the sexual selection, a female chromosome is selected by tournament selection of size t from the female group. The selection of the male chromosome is based on the HD from the female chromosome, FV or AG of the male chromosome. Detailed description can be found in [20].

To justify the decision of changing the layout of the male and female group in each generation, we tested this technique on four nonlinear numerical functions and compared it with some other sexual selection mechanism available in the literature. A standard GA is used in this experiment with a population size of 20, uniform crossover with probability, $p_c = 1.00$, and binary mutation with probability, $p_m = 0.002$. In order to better compare the mate selection technique, two types of selection mechanisms, random and tournament selections, are considered. The different sexual selection mechanisms collected from the literature that are used in the GA are listed in Table 1. The computations are to maximise the four test functions listed in Table 2 [22]. Each test function is tested on the GA for 30 times with 2000 generations per run.

The average results of 30 runs of the test functions are listed in Table 3. By considering the results from each test function, it is clear that the proposed technique of grouping the male and female chromosomes alternately outperforms other grouping techniques of sexual selection mechanisms.

Table 1: Sexual selection mechanisms.

Population order	Grouping of male and female	Selection	Selection mechanism
Randomly scattered	First half male, second half female	1	Male and female randomly
		2	Male and female with tournament size two
	Select male and female alternately	3	Male and female randomly
		4	Male and female with tournament size two
Sorted in non-increasing order of fitness values	First half male, second half female	5	Male and female randomly
		6	Male and female with tournament size two
	Select male and female alternately	7	Male and female randomly
		8	Male and female with tournament size two

Table 2: Test functions (source: [22]).

Test Function	Constraints
$f_1(x) = \{1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \cdot \{30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$	$-10 \le x_i \le 10$
$f_2(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$-3 \le x_1 \le 3$ $-2 \le x_2 \le 2$
$f_3(x) = \{\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\} \cdot \{\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\}$	$-10 \le x_i \le 10$
$f_4(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - 1/8\pi)\cos(x_1) + 10$	$-5 \le x_1 \le 10$ $0 \le x_2 \le 15$

Table 3: Comparison of gender grouping mechanisms.

	f_1	f_2	f_3	f_4
Selection 1	7.02×10^{10}	121.56	133.82	267.57
Selection 2	8.09×10^{10}	**162.20**	210.08	**308.13**
Selection 3	8.09×10^{10}	**162.20**	210.08	308.13
Selection 4	$\mathbf{8.35 \times 10^{10}}$	**162.20**	**210.23**	**308.13**
Selection 5	7.02×10^{10}	110.21	98.66	221.55
Selection 6	7.49×10^{10}	117.02	60.14	308.08
Selection 7	8.11×10^{10}	**162.20**	209.83	**308.13**
Selection 8	$\mathbf{8.35 \times 10^{10}}$	**162.20**	**210.23**	**308.13**

3.4. Crossover

3.4.1. Fuzzy Crossover Operator

The performance of a GA is dependent on the genetic operators particularly on the type of crossover operator. Effective crossover in GAs is achieved through establishing the optimum relationship between the crossover structure and the search problem itself. The goal of this section is to provide a crossover selection technique based on the diversity of the population and a FLC. A crossover operator is selected in each generation from a list of crossover operators by running fuzzy tools on the results of measurements of the population diversity.

Table 4: Abbreviations of crossover operators.

Crossover operators	Abbreviation
2-Point crossover	2PC
k-Point crossover	KPC
Uniform crossover	UC
Segregation crossover	SC
Inversion crossover	IC

Many published works compared the various crossover operators (see, De Jong and Spears [23, 24], Syswerda [25], and Eshelman et al. [26]). These researchers particularly investigated the numbers of crossover points, when some segments were exchanged using k-point type of cross-over techniques. Their results demonstrated that sometimes the final output would be better if the number of crossover points was increased. Besides empirical analysis, substantial efforts have been invested in comparing, from theoretical perspectives, mutation, and crossover as well as the various crossover operators [26]. However, these theories are not general enough to predict when to apply crossover or what type of crossover operator to employ. For instance, these theories did not address the issue of size of the study population even though it may affect the relative usefulness of the crossover operators [23]. Likewise, there is enough evidence suggesting that the relative usefulness of mutation may be influenced by the size of the population. Mutation seems to be more advantageous compared to crossover with small population sizes whereas crossover may be more advantageous than mutation when dealing with large population sizes [26]. Moreover, the relative usefulness of mutation and crossover are influenced by several other factors like the fitness function, the representation, and the selection scheme.

For selecting the most appropriate crossover method, the structure of the crossover and the population diversity should be considered. In this study, we are using the technique introduced in [20]. In this technique, some crossover operators based on binary encoding are considered as the default crossover operators. These methods and their abbreviations are presented in Table 4.

(i) Two-Point Crossover [2]

In a two-point crossover operator, two positive integer random numbers r_1 and r_2 are generated as cross-sites such that $r_1, r_2 < L$, where L is the length of the chromosome. The genes bracketed by these numbers are exchanged between two mated parents. If the number of crossover point is greater than two, this type of crossover is named as k-Point crossover. In a k-Point crossover, a positive integer random number, r, is generated such that $r \leq L$, where r is the number of cross-points. The genes are exchanged between alternate pairs of the cross-points.

(ii) Uniform Crossover [25]

In the uniform crossover operator, each gene from either parent is selected for exchange and an offspring is created by copying the corresponding gene from one of the parent which is chosen using a randomly generated crossover mask. In each generation, a new crossover mask is randomly generated for each pair of parents.

Table 5: Levels of crossover operator based on CA (source: [20]).

Low	Medium	High
2-point	k-point Uniform	Segregation Inversion

(ii) Segregation Crossover [27]

In the segregation crossover operator, two cuts, one from each parent, are selected so that the lengths of these parts are the same and then all genes in these parts are exchanged. In this technique, similarity in the gene positions for exchange is not a necessity. The difference between the SC and 2PC methods lies in the positions of the genes that are exchanged.

(iii) Inversion Crossover [27]

In this technique, two positive integer random numbers r_1 and r_2 are generated as cross-sites such that $r_1, r_2 < L$. The contents bracketed by these sites are exchanged between two parents. Then, the end points of these exchanged contents switch places.

The explanation of the relative crossover operator and population diversity can be found in [20]. There are three levels for crossover ability (CA): low, medium and high. When the CA is low, this means that the operator is weak in keeping population diversity. Medium CA means the operator is appropriate for keeping genetic diversity and it is effective for finding different fitness value. If the CA is high, this means that the operator is useful for increasing genetic diversity and finding different fitness value. Levels of crossover operator based on CA used in this study are given in Table 5 [20].

As for the $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$ inputs, the set of linguistic labels is associated with the ability of keeping the genetic diversity of low, medium and high. For each linguistic term, there is a triangular fuzzy set that defines its semantic (meaning) as shown in Figure 2. Note that the standard Mamdani method is used for fuzzy inference system.

3.4.2. Fuzzy Crossover Probability

Crossover is based on probabilistic model-building procedure in a GA. It controls the rate at which solutions are to crossover given that, for each problem, the probability is chosen by the user. The selection of the crossover probability, p_c, critically affects the behaviour and performance of a GA. However, guiding selection principles exist in the literature, (see, Grefenstette [28], De Jong [29, 30], Goldberg and Sastry [31], and Schaffer et al. [32]). Researchers usually use a probability of 50%–100% for executing the crossover [33]. Goldberg and Sastry [31] generalized the schema theorem for p_c. They argued that the selection pressure corresponds to disruption of schema and showed that when the building blocks are compact (meaning its genes are located close to each other in the chromosome string), the GA works well for a wide range of combinations of p_c and selection rates.

Adaptive techniques that alter the probability of applying an operator in proportion to the observed performance of the chromosomes created by that operator in the course of a run were proposed by Davis [34]. Fernandes et al. [35] considered a GA that adapts the reproduction rate to the size of the population under investigation. Annunziato and

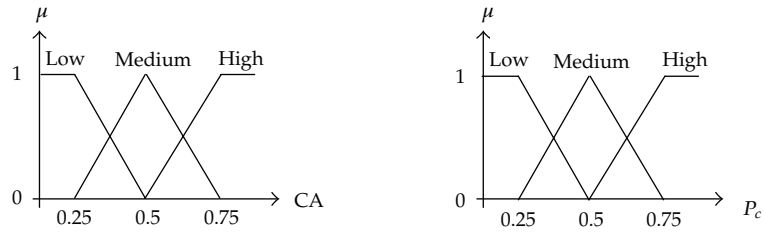

Figure 2: The set of linguistic labels associated with CA and p_c (source: [20]).

Table 6: Abbreviations of mutation operators.

Mutation operator	Abbreviation
Binary mutation	BM
Interchanging mutation	IM
Reversing mutation	RM
Parity Encoding mutation	PEM
Simple Sum Coding mutation	SSCM
Inversion Sum Coding mutation	ISCM
Cycle Sum Coding mutation	CSCM

Pizzuti [36] presented a technique for setting the genetic parameters during the course of a run by adapting the population size and the operator rates to the environmental constraint of maximum population size. On the other hand, an approach to determine of the probability of crossover and mutation based on the various fitness potentials of the population was established by Srinivas and Patnaik [18] where they presented a particular approach which eliminates the need for parameters in a crossover-based GA.

In this study, the crossover probability is varied on the basis of the phenotype and genotype characteristics of the chromosome. In addition, the population diversity is considered and its probability is estimated by an FLC. As for the $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$ inputs, the set of linguistic labels associated with crossover probability comprises of the descriptions of low, medium, and high. For each linguistic term, there is a triangular fuzzy set that defines its semantic (meaning) as shown in Figure 2 [20].

3.5. Mutation

3.5.1. Fuzzy Mutation Operator

In GAs, a mutation operator is applied after performing the crossover. The mutation operator is used to prevent the GAs from falling into a local optimum by preventing the population of chromosomes from becoming very similar to each other, which results in a premature convergence. The arguments presented in this subsection lead to the idea of using fuzzy system for selecting an appropriate mutation operator in order to obtain a more efficient mutation operator. In this study, some of the mutation operators are chosen from the literature. The Common abbreviations of these methods are summarized in Table 6.

(i) Binary Mutation [2]

A binary mutation operator is applied to each offspring which is represented by a binary code. It introduces small changes to the individual by altering each gene in the offspring from "1" to "0" or vice versa.

(ii) Interchanging Mutation [37]

In an interchanging mutation, two random position of the string are chosen and the bits corresponding to those positions are interchanged.

(iii) Reversing Mutation [37]

In a reversing mutation, a random position is chosen and the bits next to that position are reversed to produce a new offspring.

(iv) Parity Encoding Mutation [38]

Let $X = \{x_k\}$ be an n-bit binary sequence, indexed from left to right of chromosome c_i, the parity encoding, $P(X) = \{y_j\}$, of X is defined to be $y_j = \sum x_k,\ k = 1$ to j where $\sum$ is the base 2 summation. For example, $P(1101001) = 1001110$ and $P(1001110) = 111010$. In the PEM, X is replaced by $P(X)$ in offspring c_i.

(v) Simple Sum Coding Mutation [38]

In this operator, a substring of offspring is selected randomly and then this subset is summed on a base 2 and is replaced with the same place in the offspring. If the number of genes after summation is more than the places, the extra genes are not considered (right to left). For example, consider a substring of offspring p_i = (10**1100**11100001011) that is chosen between the second and 7th gene. This result is (1100) + (1100) = 11000 and 1000 replaced the 1100 and new offspring is p_i' = (10**1000**11100001011).

(vi) Inversion Sum Coding Mutation [38]

A substring of offspring is selected randomly, and then the subset is summed with inverse order of itself on base 2 and replaced with the same place in the offspring. If the number of genes after summation is more than the places, the extra genes are not considered (right to left). For example, consider a substring of offspring p_i = (10**1100**11100001011) that is chosen between the second and 7th gene. This result is (1100) + (0011) =1111 and 1111 replaced the 1100 and new offspring is p_i' = (10**1111**11100001011).

(vii) Cycle Sum Coding Mutation [38]

In this operation, the start of the sequence is connected to the end to form a cycle. In this cycle, two substrings are considered where the lengths are the same. These substrings are summed together on a base 2 and the result replaces the position of one of the substrings randomly. If the number of genes after summation is more than the places, the extra genes are not considered (right to left).

Mutation:	Before	After
Offspring 1:	1 0 1 0 1 **0 1 0 1 0**	1 0 1 0 1 **0 1 0 1 0**
Offspring 2:	1 1 1 0 0 0 **1 1 1 1**	1 1 1 0 0 0 **1 1 1 1**

Figure 3: An example of Reversing Mutation.

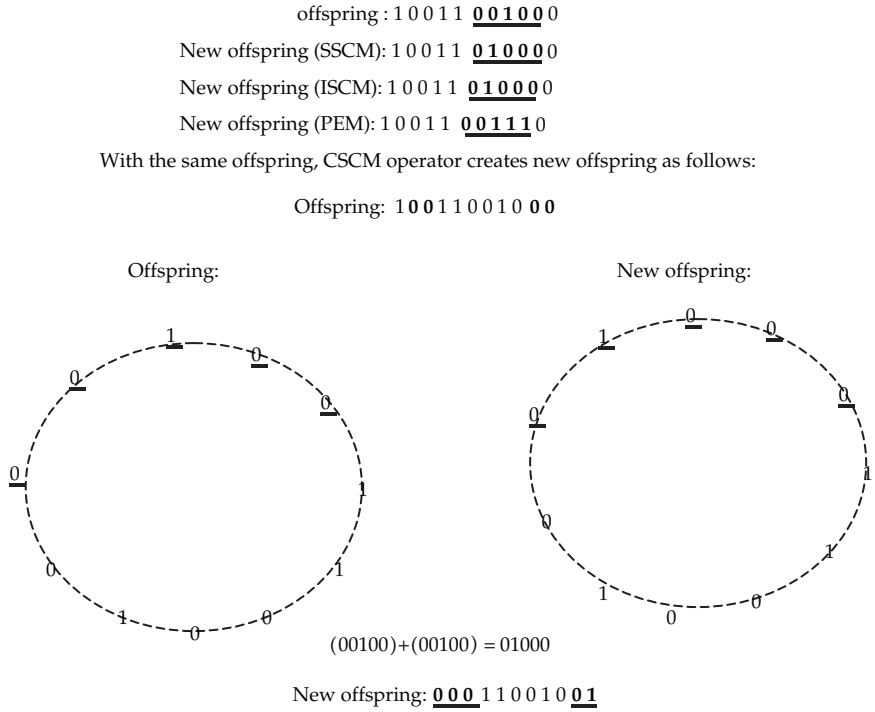

Figure 4: Comparison of mutation operators of on offspring.

When the BM operator is used as a default mutation operator, a small mutation probability such as $p_m = 1/L$, where L is the length of the chromosome, is used as the most common approach. However, this technique has two shortcomings:

(i) when the length of chromosome is short, the mutation probability is high which forces us to choose an experimentation strategy for the p_m selection,

(ii) each gene will be checked based on the mutation probability which renders a high computation time.

In the IM operator, only two genes are changed for each chromosome. Note that for a long chromosome, this method is not appropriate. If the layout of genes is symmetric or if all genes are similar, the RM technique failed to create new offspring as shown by the example in Figure 3.

When the SSCM, ISCM, CSCM, or PEM is used as the mutation operator, the performances of the CSCM and PEM operators are sometimes better than those of the ISCM and SSCM operators (Figure 4).

Table 7: Levels of mutation operator based on MA.

Low	Medium	High
Interchanging	Binary	Parity Encoding
Reversing	Simple Sum Coding	Inversion Sum Coding Cycle Sum Coding

When the genetic diversity of the population is high ($T_{3,t} \geq 0.50$), the IM and RM operators perform well because the genetic diversity is high and there is no need to made huge changes. Also the time consumptions in the IM and RM methods are less than other operators. Therefore, it is better to use IM or RM in this case. The diversity of the population is considered medium when some gene locations in the chromosomes are the same ($0.25 \leq T_{3,t} \leq 0.50$). In such cases, it is better to use mutation operators that can keep the diversity or amplitude changes on the genetic diversity. Therefore, it is better to use BM or SSCM operators in this case. On the other hand, the diversity of the population is considered low ($T_{3,t} < 0.25$) when some chromosomes are identical. Hence, we need to use mutation operators that can increase the genetic diversity of the population but in reasonable computational time. IM and RM are weak for this situation. In this case, we can use BM or SSCM operator, but the problem is that we have a high mutation probability. Therefore, it is better to use PEM, ISCM, or CSCM in this case. The explanation of the relative mutation operators with the population diversity which consist of three levels of mutation ability (MA) is given in Table 7.

As for the $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$ inputs, the set of linguistic labels associated with MA for keeping genetic diversity comprises the descriptions of low, medium and high. For each linguistic term, there is a triangular fuzzy set that defines its semantic (meaning) as shown in Figure 5.

3.5.2. Fuzzy Mutation Probability

The efficiency of the mutation operator as a means of exploring the search space is questionable. A GA using mutation as the only genetic operator would be a random search that is biased toward sampling good hyperplanes rather than poor ones. In order to explore the space reasonably quickly, the p_m needs to be high. However, a high p_m would disturb the sampling rates, even for the short schemata, and we would then end up with an essentially pure random search. On the other hand, if p_m is low, the search would be very slow since application of the mutation operator would often leave a string unchanged. Then the question is: how could one figure out the probability value to be used for mutation?

This study has taken the population diversity into consideration and the related probability of mutation has been calculated using an FLC. As for the $T_{1,t}$, $T_{2,t}$, and $T_{3,t}$ inputs, the set of linguistic labels associated with mutation probability comprises the descriptions of low, medium and high. For each linguistic term, there is a triangular fuzzy set that defines its semantic (meaning) as shown in Figure 5.

In a traditional GA, usually a very small number is used for mutation probability. Most of them used $p_m = 1/L$. In this proposed technique, $p_m = 1/L$ is used as the center point for the mutation probability. The range that is considered in this study is at the interval of $(1/L - 1/2L, 1/L + 1/2L) = (1/2L, 3/2L)$.

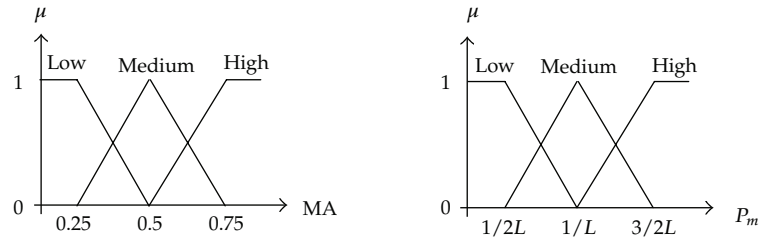

Figure 5: The set of linguistic labels associated with MA and p_m.

3.6. Fuzzy Rules

The linguistic rules describing the control system consist of two parts: an antecedent block (between the **IF** and **THEN**) and a consequent block (following **THEN**). By making this type of evaluation, fewer rules can be evaluated, thus simplifying the processing logic and perhaps even improving the fuzzy logic system performance. There is the possibility of generating a single rule for each output variable. In this paper, we used the toolbox of Mamdani inference systems from MATLAB fuzzy logic toolbox. The inputs are combined logically using the AND/OR operator to produce output $(x_i,\ \mu_i(\mathrm{CA}))$, $(y_i,\ \mu_i(\mathrm{MA}))$, $(z_i,\ \mu_i(p_c))$, and $(w_i,\ \mu_i(p_m))$ response values for all the expected inputs [20]:

$$\begin{aligned}
\mu_i(\mathrm{CA}) &= \max\{\mu_i(T_{1,t}), \mu_i(T_{2,t}), \mu_i(T_{3,t})\},\\
\mu_i(\mathrm{MA}) &= \max\{\mu_i(T_{1,t}), \mu_i(T_{2,t}), \mu_i(T_{3,t})\},\\
\mu_i(p_c) &= \min\{\mu_i(T_{1,t}), \mu_i(T_{2,t}), \mu_i(T_{3,t})\},\\
\mu_i(p_m) &= \min\{\mu_i(T_{1,t}), \mu_i(T_{2,t}), \mu_i(T_{3,t})\},
\end{aligned} \tag{3.3}$$

where $i = 1, 2, \ldots,$ the number of rules.

Therefore, the proposed fuzzy system with three input variables will have 18 ($3 \times 2 \times 3$) rules for each output variable (CA, MA, p_c, and p_m). This fuzzy rule base is collectively presented in Table 8. The fuzzy outputs for all rules are finally aggregated to one fuzzy set. To obtain a crisp decision from this fuzzy set, we use the centre of gravity approach for defuzzification [20]:

$$\begin{aligned}
x_{\mathrm{CA}} &= \frac{\sum_{i=1}^{n} \mu_i(\mathrm{CA}) x_i}{\sum_{i=1}^{n} \mu_i(\mathrm{CA})},\\
y_{\mathrm{MA}} &= \frac{\sum_{i=1}^{n} \mu_i(\mathrm{MA}) y_i}{\sum_{i=1}^{n} \mu_i(\mathrm{MA})},\\
z_{p_c} &= \frac{\sum_{i=1}^{n} \mu_i(p_c) z_i}{\sum_{i=1}^{n} \mu_i(p_c)},\\
w_{p_m} &= \frac{\sum_{i=1}^{n} \mu_i(p_m) w_i}{\sum_{i=1}^{n} \mu_i(p_m)}.
\end{aligned} \tag{3.4}$$

Table 8: Fuzzy rule base for CA, p_c, MA, and p_m.

Rule	$T_{1,t}$	$T_{2,t}$	$T_{3,t}$	CA	p_c	MA	p_m
1	Low	Low	Low	High	High	High	High
2	Low	Low	Medium	High	High	High	High
3	Low	Low	High	Medium	Medium	Medium	High
4	Low	High	Low	High	Medium	High	High
5	Low	High	Medium	Medium	Medium	Medium	Medium
6	Low	High	High	Medium	Low	Medium	Low
7	Medium	Low	Low	High	High	High	High
8	Medium	Low	Medium	Medium	Medium	Medium	Medium
9	Medium	Low	High	Medium	Medium	Medium	Low
10	Medium	High	Low	Medium	Medium	Medium	Medium
11	Medium	High	Medium	Medium	Medium	Medium	Low
12	Medium	High	High	Medium	Low	Medium	Low
13	High	Low	Low	High	High	High	High
14	High	Low	Medium	Medium	Medium	Medium	Medium
15	High	Low	High	Low	Medium	Low	Low
16	High	High	Low	Low	Medium	Low	Medium
17	High	High	Medium	Low	Low	Low	Low
18	High	High	High	Low	Low	Low	Low

3.7. Elitism Replacement with Filtration

After the fuzzy crossover and mutation operators are applied, elitism replacement technique is used as the replacement strategy. The offsprings will have to compete with their parents in order to allow transition into the new population. In other words, fitter chromosomes will survive for the next generation and they are never lost unless better solutions are found. In the elitism replacement technique, both parent and offspring populations are considered together as a single population. Then, this population is sorted in a nonincreasing order of their associated fitness value and the first half of the chromosomes from this combined population are selected as the chromosomes of the new population for the next generation.

What is meant, within this context, by saying that two chromosomes are identical is that the loci of their genes are equal. Existence of the identical chromosomes is one of the critical problems for premature convergence. When large proportion of the chromosomes in the population are identical, the diversity of the population will be lost and premature convergence occurs. In order to overcome this problem, the filtration technique is used to add diversity to the new population. In this technique, one of the identical chromosomes is kept while the others are removed and replaced by new feasible chromosomes that are generated randomly. As the filtration procedure involves the process of "identifying," "regenerating" and "reevaluating," of the new chromosomes which requires a fair amount of computation time, it is sensible to just invoke the procedure every R generation (where R is a parameter, e.g., 100) or when there is at least 10 percent of the population that are identical [20].

4. Computational Experiments

The multidimensional knapsack problem (MKP) is defined as a combinatorial optimization problem that is NP hard. The MKP problems have large application, which includes many

applicable problems such as cargo loading, cutting stock, and bin-packing, among others. The goal of an MKP is to boost the sum of values of the items to be chosen from some specified set by means of taking multiple-resource restraints into consideration. This problem has been widely studied over many decades due both to theoretical interests and its broad applications in several fields, like engineering, operations research, management, and computer sciences. Elaborate literature on the MKP and its relations to different problems are published by Pisinger [33], Gavish and Pirkul [39], Martello and Toth [40], Chu and Beasley [41], and Freville and Plateau [42]. Basically, the MKP can be formulated as follows [43].

$$\begin{aligned} \max \quad & f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} p_j x_j \\ \text{s.t} \quad & \sum_{j=1}^{n} w_{ij} x_{ij} \leq c_i \quad i = 1, 2, \ldots, m, \\ & x_j \in \{0, 1\} \quad j = 1, 2, \ldots, n, \\ & \text{with } p_j > 0,\ w_{ij} \geq 0,\ c_i \geq 0, \end{aligned} \tag{4.1}$$

where n: number of objects; m: number of knapsacks; w_{ij}: consumption of resource i for object j; c_i: capacity of the ith knapsack; p_j: profit associated with object j; and x_j: decision variable with object j.

In this section, we present the computational results of the proposed FGA and the comparisons with other variants of heuristic and local search methods. The remaining parts of this section are organised as follows:

(1) The proposed sexual selection is compared with some selection mechanisms from the literature.

(2) The proposed FGA based on the crossover operator and probability selection technique is compared with some crossover operators with fixed probability from the literature.

(3) The proposed FGA based on the mutation operator and probability selection technique is compared with some mutation operators with fixed probability from the literature.

(4) The completed FGA (based on sexual selection, crossover, and mutation operators with their probability selection technique) is compared with some heuristic and local search algorithms from the literature, namely; GA by Chu and Beasley [41], simulated annealing by Qian and Ding [44], and heuristic algorithms by Magazine and Oguz [45], Volgenant and Zoon [46], and Pirkul [47].

4.1. Experimental Design

The benchmark dataset tested in this research is inclusive of 270 problem instances on the multidimensional knapsack problems as proposed by Chu and Beasley [41]. These problem instances have been extensively utilized in literature for the testing of MKP algorithms. These problems include $n = 100, 250, 500$ variables and $m = 5,\ 10,\ 30$ constraints. For each category of n variables, three tightness ratios are considered as $\alpha = 0.25,\ 0.50,\ 0.75$. This set

Table 9: Abbreviation of selection mechanism.

Selection mechanism	Abbreviation
Sexual selection	SX
Roulette Wheel	RW
Tournament selection	TS
Linear Ranking	LR
Stochastic Universal Sampling	SUS
Truncation selection	TR

of problems contains 27 different problem sets, each having 10 randomly generated instances, thus a total of 270 problems.

In these problem sets, w_{ij} are drawn from discrete uniform generator $U(0, 1000)$ and the right hand side coefficients c_i, $i \in \{1, \ldots, m\}$, are set using $c_i = \alpha \sum_{j=1}^{n} w_{ij}$ where α is the tightness ratio. The objective function coefficients p_j, $j \in \{1, \ldots, m\}$ are correlated to w_{ij} and are generated as $p_j = \sum_{i=1}^{m} (w_{ij}/m) + 500e_j$, where e_j is a real number drawn from the continuous uniform generator $U(0, 1)$ [41]. The performance of these approaches is measured using the Percentage Deviation (PD) between the benchmark problems taken from the OR Library [48] and our results. That is,

$$\text{PD} = \frac{|B_i - R_i|}{B_i} \times 100\%, \tag{4.2}$$

where B_i is the benchmark result and R_i is the computational result obtained by the proposed algorithm.

All algorithms were coded in C++ and ran on a Pentium IV with 2.0 GHz CPU and 2.0 GB of RAM. For each problem instance, 30 runs were performed. In order to have a fair comparison between the algorithms in this experiment, we employed duration of 20 CPU seconds per run.

4.2. Computational Results of Selection Mechanisms

In this subsection, we present the computational results of the proposed sexual selection and the comparisons with other variants of selection mechanisms from the literature. The abbreviations of these methods are presented in Table 9.

The computational experiments reported here were performed using standard GAs (SGAs) with the initial population technique that is proposed by Chu and Beasley [41]. In these algorithms, the population size of 100, uniform crossover with probability, $p_c = 0.70$, and binary mutation with probability, $p_m = 1/L$, are considered. The replacement strategy is the technique explained in Section 3.7.

Generally, the classical selection mechanisms are trying to keep better chromosomes in the population and consequently they tend to be biased toward certain genes. This would lead to premature convergence. In order to prevent premature convergence, diverse solutions and a good coverage of the solution landscape have to be provided in the population. To see whether the sexual selection provides a good coverage and to avoid the premature convergence, in the Figure 6, there are the average PDs computed over 10 problem instances

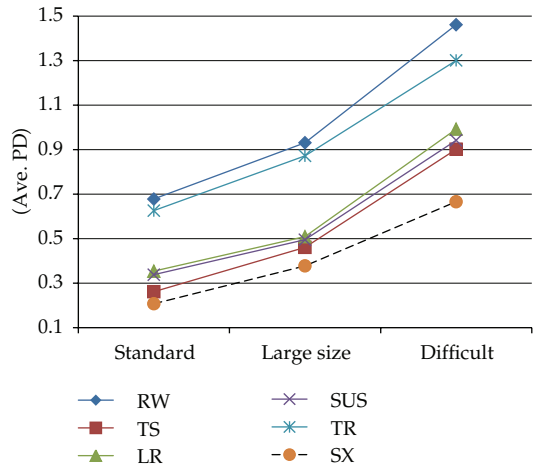

Figure 6: Average percentage deviation between three categories of standard, large size, and difficult MKPs based on selection mechanisms.

with 30 runs each for the selection mechanisms mentioned in Table 9 when the algorithms terminate at 20 CPU seconds per run.

Results emphasised in Figure 6 show that the sexual selection finds better and comparable solutions for three categories of standard, large size and more difficult problems described in [41]. Hence, the sexual selection is effective for keeping population diversity, avoiding premature convergence, and, consequently, finding better and comparable solutions.

4.3. Computational Results of FGA Based on the Crossover Operator and Probability Selection Technique

In this subsection, we present the computational results of the proposed FGA based on the crossover operator and probability selection technique named FGASC for solving the MKPs. In order to scrutinize the performance of the proposed FGASC, some commonly used crossover operators in binary encoding are considered. Common abbreviations of these operators were mentioned in Table 4.

The computational experiments reported here were performed by using SGAs with the initial population technique described in [41]. Once again, the population size is kept as 100, with the tournament selection of size two being used as the default selection mechanism, and the binary mutation with probability, $p_m = 1/L$, is considered. The replacement strategy is the technique explained in Section 3.7. The crossover operators listed in Table 4 are applied with a fixed crossover probability, $p_c = 0.70$ on the SGAs. The average PDs computed over 10 problem instances with 30 runs each of the SGAs and the FGASC when the algorithms terminate at 20 CPU seconds per run are shown in Figure 7.

Figure 7 presents the average PDs of SGAs and FGASC for three categories of standard, large size, and difficult MKPs. The results show that the FGASC performs significantly better than the SGAs. This shows that the FGASC is able to produce better quality solutions compared to SGAs in a fixed CPU time. There is clear evidence from Figure 7

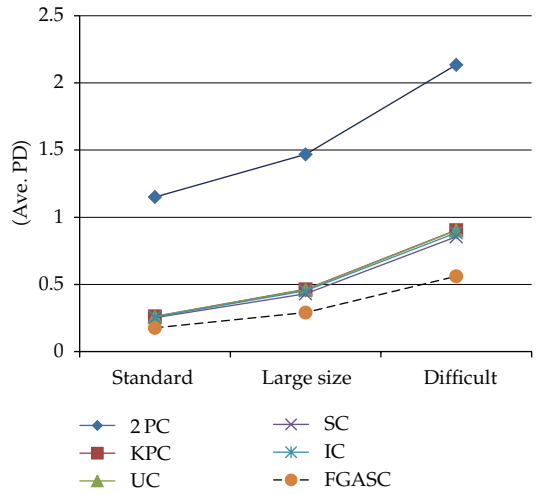

Figure 7: Average percentage deviation between three categories of standard, large size, and difficult MKPs based on crossover operators.

that, on average, the FGASC is the best algorithm followed by the SGAs with SC, IC, UX, KPC, and finally the SGA with 2PC. The algorithms find problems of $m = 30$ to be the most challenging. With other things (n and α) being equal, when we increase the value of m, then PD will increase.

4.4. Computational Results of FGA Based on the Mutation Operator and Probability Selection Technique

The computational results of the proposed FGA based on the mutation operator and probability selection technique called FGASM is presented in this subsection. In order to assess the performance of the proposed FGASM, some commonly used mutation operators in binary encoding mentioned in Table 6 are considered.

The computational experiments reported here were performed by using SGAs with the initial population technique introduced by [41]. In these algorithms, a population with size of 100, tournament selection of size two, 2-point crossover with fixed probability, $p_c = 0.70$, and a fixed mutation probability, $p_m = 1/L$, are considered. A 2-point crossover is used because this operator is weak in keeping the population diversity. Hence, the effect of the mutation operators towards the computational results will be more comparable. The replacement strategy is technique explained in Section 3.7. The average PDs of the comparison of the FGASM with the other mutation operators on SGAs are shown in Figure 8.

Figure 8 presents the average PDs computed over 10 problem instances with 30 runs each of the SGAs and FGASM for three categories of standard, large size, and difficult MKPs. As observed from Figure 8, the highest average PD is obtained by RM. The average PDs for SSCM, ISCM, CSCM, and PEM are better than that of BM, IM, and RM whereas the average PDs for FGASM are the best. Therefore, we can conclude that the mutation operator and probability selection technique are effective in keeping population diversity, avoiding premature convergence, and, consequently, finding better and comparable solutions for the MKPs.

Table 10: Abbreviation of the heuristics and local search algorithms.

Algorithm	Abbreviation
Heuristic by Magazine and Oguz	M&O
Heuristic by Volgenant and Zoon	V&Z
Heuristic by Pirkul	MKHEUR
Simulated Annealing by Qian and Ding	SA
GA by Chu and Beasley	GACB
Fuzzy Genetic Algorithm	FGA

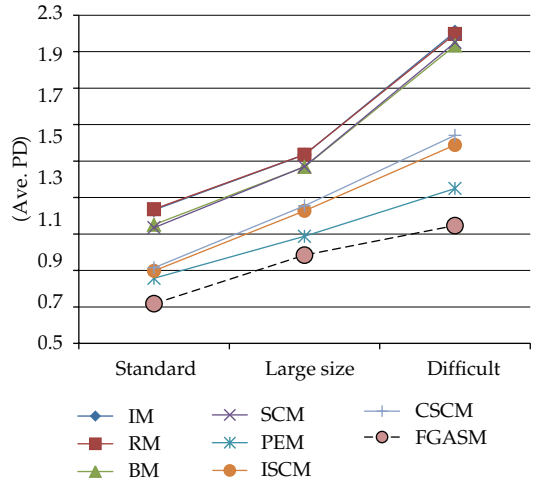

Figure 8: Average percentage deviation between three categories of standard, large size, and difficult MKPs based on mutation operators.

4.5. Comparison of FGA with Heuristics and Local Search Algorithms

In this subsection, we present the results of the proposed complete FGA for solving the MKPs. In order to assess of the performance of the proposed FGA, the heuristics and local search algorithms found in the literature are considered. The abbreviations of these methods are presented in Table 10.

For all these algorithms, since the original source codes were not available to us, we used the computational results presented in Chu and Beasley [41] and Qian and Ding [44]. The same benchmark problem instances are used for the computational experiments. The numerical results are computed after making 30 independent runs for statistical significance. Each run is terminated if one of the following stopping conditions is satisfied:

(1) the fitness value does not improved after 100 generations;

(2) the number of generations is equal to 10^6; or

(3) the CPU time is more than 500 CPU seconds.

Table 11 directly compares the performance of the complete FGA with other well-known heuristics and local search algorithms by means of average PDs. The benchmark column represents the average computational results of 10 problem instances from each problem set

Table 11: Computational results of FGA with heuristics and local search algorithms.

m	n	α	Benchmark	Heuristics and local search algorithms					FGA
				M&O	V&Z	MKHEUR	SA	GACB	
5	100	0.25	24197.20	13.69	10.30	1.59	2.95	0.99	0.12
		0.50	43252.90	6.71	6.90	0.77	1.67	0.45	0.08
		0.75	60471.00	5.11	5.86	0.48	0.87	0.32	0.01
	250	0.25	60409.70	6.64	5.85	0.53	3.09	0.23	0.01
		0.50	109284.60	5.22	4.40	0.24	1.89	0.12	0.00
		0.75	151555.90	3.56	3.59	0.16	0.84	0.08	0.00
	500	0.25	120615.50	4.93	4.11	0.22	3.41	0.09	0.00
		0.50	219503.10	2.96	2.53	0.08	2.04	0.04	0.00
		0.75	302354.90	2.31	2.41	0.06	0.92	0.03	0.00
			Average	**5.68**	**5.12**	**0.46**	**1.96**	**0.26**	**0.02**
10	100	0.25	22601.90	15.88	15.55	3.43	3.93	1.56	0.23
		0.50	45659.10	10.41	10.72	1.84	2.09	0.79	0.11
		0.75	59555.60	6.07	5.67	1.06	1.06	0.48	0.02
	250	0.25	58993.90	11.73	10.53	1.07	3.21	0.51	0.02
		0.50	108706.40	6.83	5.92	0.57	2.14	0.25	0.01
		0.75	151330.40	4.42	3.77	0.33	1.01	0.15	0.00
	500	0.25	118565.50	8.81	7.90	0.52	3.67	0.24	0.00
		0.50	217274.60	5.71	4.14	0.22	2.50	0.11	0.00
		0.75	302556.00	3.62	2.93	0.14	1.17	0.07	0.00
			Average	**8.16**	**7.46**	**1.02**	**2.31**	**0.46**	**0.04**
30	100	0.25	21654.60	17.39	17.21	9.02	4.94	2.91	0.53
		0.50	41431.30	11.82	10.19	3.51	2.62	1.34	0.28
		0.75	59199.10	6.58	5.92	2.03	1.29	0.83	0.06
	250	0.25	56875.90	13.54	12.41	3.70	3.60	1.19	0.24
		0.50	106673.70	8.64	7.12	1.53	2.25	0.53	0.05
		0.75	150443.50	4.49	3.91	0.84	0.96	0.31	0.01
	500	0.25	115473.50	9.84	9.62	1.89	3.75	0.61	0.05
		0.50	216156.90	7.10	5.71	0.73	2.30	0.26	0.00
		0.75	302353.40	3.72	3.51	0.48	1.07	0.17	0.00
			Average	**9.24**	**8.40**	**2.64**	**2.53**	**0.91**	**0.14**
Average				**7.69**	**6.99**	**1.37**	**2.27**	**0.54**	**0.07**

of the MKPs that are compiled in [48]. For each algorithm, the entries report the average PD computed over 10 problem instances with 30 runs each, which is 300 runs. For each category of m, the final line gives the average PD over all values of n. The final line of Table 11 gives the overall average PD over all categories of m. The results acquired by the FGA show that the FGA is very effective for the three categories of standard, large size, and difficult MKPs. The reported PDs are a measure of how close the heuristic solution is to the linear programming optimum; therefore, much smaller values are expected if compared to the integer optimum. This can be partially verified by comparing the average percentage deviation columns given in Table 11. Finally, the ability of the FGA to generate optimal solutions is demonstrated in the last column, in which the FGA is able to find optimal values for many instances.

Table 11 obviously indicates the transcendent of our proposed FGA over other heuristics and local search algorithms in terms of the quality of the solutions obtained.

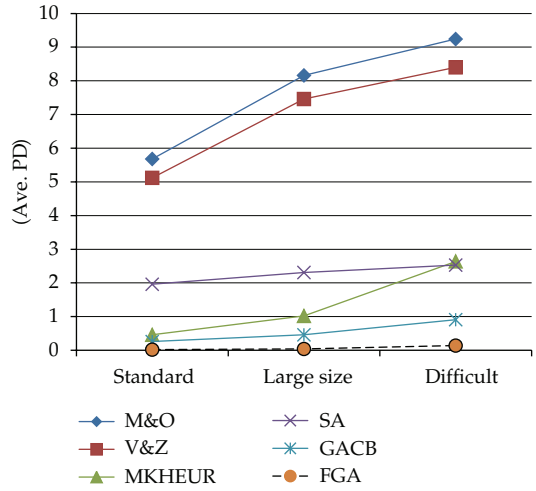

Figure 9: Average percentage deviation of FGA with heuristics and local search algorithms.

The average PDs between our best solution and the best-known solution of benchmark is between 0.00% and 0.14% whereas for other heuristics and local search algorithms are 0.26% and 0.91%. On the other hand, if we consider the average PDs for the three categories of standard, large size, and difficult MKPs together, the best average PDs for heuristics and local search algorithms is 0.543%, whereas this number for FGA is 0.068%.

From another point of view, as can be seen in Figure 9, the M&O and V&Z have the highest average PDs whereas the average PDs for SA, MKHEUR, and GACB algorithms are better than that of the M&O and V&Z algorithms with GACB being the best algorithm between them. However, Figure 9 shows the proposed FGA generates solutions that have much smaller average PDs than other algorithms in all cases. All the algorithms find problems of $m = 30$ to be the most challenging. With other things (n and α) being equal, when we increase the value of m, then PD will increase. Note that for all the algorithms, fewer generations are executed within the time limit as the number of knapsacks or constraints become larger.

5. Conclusion

This paper proposed some techniques for the measurement of the population diversity based on the phenotype and genotype properties. The contributions of this study are the sexual selection, the crossover and mutation operators, and probabilities selection techniques based on the population diversity using fuzzy logic controllers. The proposed fuzzy genetic algorithm is compared to other heuristic and local search algorithms for solving the multidimensional 0/1 knapsack problems. Extensive computational experiments are carried out to assess the effectiveness of the proposed algorithms compared to other metaheuristic proposed in the literature. The computational results showed that the proposed sexual selection and FGAs are competitive and capable of generating near optimal solutions.

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, Calif, USA, 1979.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[3] F. Herrera and M. Lozano, "Fuzzy genetic algorithms: issues and models," Tech. Rep. DECSAI-98116, Department of Computer Science and A.I., University of Granada, 1998.

[4] Y. H. Song, G. S. Wang, P. Y. Wang, and A. T. Johns, "Environmental/economic dispatch using fuzzy logic controlled genetic algorithms," *IEE Proceedings: Generation, Transmission and Distribution*, vol. 144, no. 4, pp. 377–382, 1997.

[5] Y. Yun and M. Gen, "Performance analysis of adaptive genetic algorithms with fuzzy logic and heuristics," *Fuzzy Optimization and Decision Making*, vol. 2, no. 2, pp. 161–175, 2003.

[6] R. Subbu, A. C. Sanderson, and P. P. Bonissone, "Fuzzy logic controlled genetic algorithms versus tuned genetic algorithms: an agile manufacturing application," in *Proceedings of the IEEE International Symposium on Intelligent Control (ISIC '98)*, pp. 434–440, September 1998.

[7] Q. Li, X. Tong, S. Xie, and G. Liu, "An improved adaptive algorithm for controlling the probabilities of crossover and mutation based on a fuzzy control strategy," in *6th International Conference on Hybrid Intelligent Systems and 4th Conference on Neuro-Computing and Evolving Intelligence (HIS-NCEI '06)*, December 2006.

[8] C. H. Wang, T. P. Hong, and S. S. Tseng, "Integrating membership functions and fuzzy rule sets from multiple knowledge sources," *Fuzzy Sets and Systems*, vol. 112, no. 1, pp. 141–154, 2000.

[9] K. Wang, "A new fuzzy genetic algorithm based on population diversity," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 108–112, 2001.

[10] H. Liu, Z. Xu, and A. Abraham, "Hybrid fuzzy-genetic algorithm approach for crew grouping," in *5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*, pp. 332–337, September 2005.

[11] F. Herrera, E. Herrera-Viedma, M. Lozano, and J. L. Verdegay, "Fuzzy tools to improve genetic algorithms," in *Proceedings of the European Congress on Intelligent Techniques and soft Computing*, pp. 1532–1539, 1994.

[12] F. Herrera and M. Lozano, "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds., pp. 95–125, Physica, 1996.

[13] F. Herrera and M. Lozano, "Fuzzy adaptive genetic algorithm: design, taxonomy, and future directions," *Soft Computing*, vol. 7, no. 8, pp. 545–562, 2005.

[14] M. A. Lee and H. Takagi, "Dynamic control of genetic algorithms using fuzzy logic techniques," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 76–83, 1993.

[15] S. M. Im and J. J. Lee, "Adaptive crossover, mutation and selection using fuzzy system for genetic algorithms," *Artificial Life and Robotics*, vol. 13, no. 1, pp. 129–133, 2008.

[16] M. Jalali Varnamkhasti and L. S. Lee, "Fuzzy genetic algorithm with sexual selection," in *Proceedings of the 2nd International Conference and Workshops on Basic and Applied Sciences and Regional Annual Fundamental Science Seminar*, pp. 38–43, 2009.

[17] S. A. Jafari, S. Mashohor, and M. J. Varnamkhasti, "Committee neural networks with fuzzy genetic algorithm," *Journal of Petroleum Science and Engineering*, vol. 76, no. 3-4, pp. 217–223, 2011.

[18] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[19] K. Q. Zhu and Z. Liu, "Empirical study of population diversity in permutation-based genetic algorithm," in *Proceedings of the 15th European Conference on Machine Learning (ECML '04)*, vol. 3103 of *Lecture Notes in Computer Science*, pp. 537–547, 2004.

[20] M. Jalali Varnamkhasti, L. S. Lee, M. R. Abu Bakar, and W. J. Leong, "A genetic algorithm with fuzzy crossover operator and probability," *Advances in Operations Research*, vol. 2012, Article ID 956498, 16 pages, 2012.

[21] C. Jassadapakorn and P. Chongstitvatana, "Diversity control to improve convergence rate in genetic algorithms," in *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL '03)*, vol. 2690 of *Lecture Notes in Computer Science*, pp. 421–425, 2004.

[22] H. M. Voigt, J. Born, and I. Santibanez-Koref, "A multivalued evolutionary algorithm," Tech. Rep. TR-93-022, International Computer Science Institute, Berkeley, Calif, USA, 1993.

[23] K. A. De Jong and W. M. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Comput. Sci.*, pp. 38–47, 1991.

[24] K. A. De Jong and W. M. Spears, "A formal analysis of the role of multi-point crossover in genetic algorithms," *Annals of Mathematics and Artificial Intelligence*, vol. 5, no. 1, pp. 1–26, 1992.

[25] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2–9, 1989.

[26] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Biases in the crossover landscape," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 10–19, 1989.

[27] S. Rajasekaran and G. A. Vijayalakshmi Pai, *Neural Networks, Fuzzy Logic and Genetic Algorithms Synthesis and Application*, PHL Learning Private Limited, New Dehli, India, 2008.

[28] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[29] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, 1975.

[30] K. A. De Jong, "Adaptive system design: a genetic approach," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 9, pp. 566–574, 1980.

[31] D. E. Goldberg and K. Sastry, "A practical schema theorem for genetic algorithm design and tuning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 328–335, 2001.

[32] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proceedings of the 3rd International Conference Genetic Algorithms*, pp. 51–60, 1989.

[33] D. Pisinger, "An expanding-core algorithm for the exact 0-1 knapsack problem," *European Journal of Operational Research*, vol. 87, no. 1, pp. 175–187, 1995.

[34] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 91–69, 1989.

[35] C. Fernandes, R. Tavares, and A. Rosa, "niGAVaPS–outbreeding in genetic algorithms," in *Proceedings of the Symposium on Applied Computing (ACM '00)*, pp. 477–482, 2000.

[36] M. Annunziato and S. Pizzuti, "Adaptive parameterization of evolutionary algorithms driven by reproduction and competition," in *Proceedings of the European Symposium on Intelligent Techniques (ESIT '00)*, pp. 246–256, 2000.

[37] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, Berlin, Germany, 2008.

[38] M. Jalali Varnamkhasti, "A genetic algorithm based on new mutation for solving 0/1 knapsack problem," *Far East Journal of Mathematical Science*, vol. 64, no. 1, pp. 23–35, 2012.

[39] B. Gavish and H. Pirkul, "Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality," *Mathematical Programming*, vol. 31, no. 1, pp. 78–105, 1985.

[40] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, UK, 1990.

[41] P. C. Chu and J. E. Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[42] A. Freville and G. Plateau, "An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem," *Discrete Applied Mathematics*, vol. 49, no. 1–3, pp. 189–212, 1994.

[43] F. Djannaty and S. Doostdar, "A hybrid genetic algorithm for the multidimensional knapsack problem," *International Journal of Contemporary Mathematical Sciences*, vol. 3, no. 9–12, pp. 443–456, 2008.

[44] F. Qian and R. Ding, "Simulated annealing for the 0/1 multidimensional knapsack problem," *Numerical Mathematics*, vol. 16, no. 4, pp. 320–327, 2007.

[45] M. J. Magazine and O. Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem," *European Journal of Operational Research*, vol. 16, no. 3, pp. 319–326, 1984.

[46] A. Volgenant and J. A. Zoon, "Improved heuristic for multidimensional 0-1 knapsack problems," *Journal of the Operational Research Society*, vol. 41, no. 10, pp. 963–970, 1990.

[47] H. Pirkul, "A heuristic solution procedure for the multiconstrained zero-one knapsack problem," *Naval Research Logistics*, vol. 34, no. 2, pp. 161–172, 1987.

[48] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.