From Fitness Landscape to Crossover Operator Choice

Stjepan Picek
Digital Security Group - ICIS
Radboud University Nijmegen
The Netherlands
stjepan@computer.org

Domagoj Jakobović
Faculty of Electrical Engineering and Computing
University of Zagreb
Croatia
domagoj.jakobovic@fer.hr

ABSTRACT

Genetic algorithms are applied to numerous problems that demonstrate different properties. To efficiently solve these problems, during the years a significant number of variation operators have been and still are created. It is a problem by itself how to correctly choose between those operators, i.e. how to find the most suitable operator (or a set) for a given problem. In this paper we investigate the choice of the suitable crossover operator on the basis of fitness landscape. The fitness landscape can be described with a number of properties, so a thorough analysis needs to be done to find the most useful ones. To achieve that, we experiment with 24 noise-free problems and floating point encoding. The results indicate it is possible to either select a suitable operator or at least to reduce the number of adequate operators with fitness landscape properties.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*

General Terms

Algorithms, Experimentation

Keywords

Heuristic Methods, Fitness Landscape, Crossover Operator, Genetic Programming, Automatic Classification

1. INTRODUCTION

Evolutionary algorithms play an important role in solving many difficult problems. To be able to solve different problems, in the last decades numerous versions of algorithms and operators have emerged that can be readily applied. Naturally, different algorithms were necessary since the "No Free Lunch" theorem states that there is no single best algorithm for all the problems [29].

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598320

However, this diversity of the algorithms, as much as it is advantageous, is often also a pitfall. It is not an easy task to choose the optimal algorithm and its parameters without a combination of experience and extensive testing. Therefore, it would be useful to have a method that can help in the choice of the parameters. In this work, we concentrate our attention to genetic algorithms with floating point encoding applied to continuous optimization. Furthermore, we are interested only in the differences in the behavior of the algorithm with respect to the choice of the crossover operator. We point out, even with this simplification, that there is still a plethora of options; for instance, in [9] there are descriptions of 89 floating point crossover operators.

To be able to give an insight about the choice of crossover operators, we analyze fitness landscape properties. The most common motivation for fitness landscape analysis is to get a better understanding of algorithm performance on a certain problem [19]. However, we approach the problem from the opposite point fo view, where we investigate fitness landscape in order to reach conclusion about the choice of the appropriate operator and, in consequence, the performance of the algorithm as a whole.

Of course, there is a question of the relative importance of crossover operator in genetic algorithms. Here, we regard crossover operator as the vital part of GA and refer to literature for a further discussion about the importance of crossover operator [18].

1.1 Related Work

In existing literature the fitness landscape analysis is used mainly for the determination of optimization problem hardness for evolutionary algorithms and other optimization algorithms, such as deterministic local search. Authors in [8] formally define fitness landscapes, provide an in-depth look at basic properties and give detailed explanations and examples of existing fitness landscape analysis techniques. A survey of the literature on fitness landscape analysis for local search is given in [26]. Fitness analysis of landscape for the n-queens problem is conducted in [13]. The distribution of local optima in this instance is uniformly random and scattered over the search space. The landscape is rugged and there is no significant correlation between fitness and distance of solutions. Landscape analysis for benchmark fitness functions: Sphere, Rastrigin, Rosenbrock and Ackley functions is performed in [14]. To better understand the influence of genetic representations and associated variation operators when solving a combinatorial optimization problem (the Optimal Golomb Ruler problem) various landscape

analysis techniques are used in [24]. The fitness landscape analysis for mutation reveals that indirect encodings have a good fitness distance correlation and distance to the optimum.

Herrera et al. apply a hybrid scheme in crossover, where each reproduction produces offspring using different crossover operators, and different parent and children selection mechanisms are used [11]. The operators are tested on 13 real-valued functions with 8 crossover operators and their pairs; performance was evaluated for individual problems, but the overall efficiency was difficult to predict [11].

1.2 Our Contribution

There are two main contributions in this paper. First, we correlate the choice of the crossover operator with the fitness landscape properties. As far as we are aware, this is the first application of fitness landscape analysis in such a way. Although there is a significant body of work about fitness landscape, usually its properties are investigated post-hoc to get a better insight in the performance of the algorithm. The second contribution is the usage of genetic programming for automatic classification of crossover eligibility on the basis of fitness landscape properties.

The rest of the paper is organized as follows: in Section 2 we give preliminary information on crossover operators and benchmark functions we consider. Additionally, we present fitness landscape properties used in this research. In Section 3 we describe experimental results on the benchmark functions. Also, we give preliminary analysis on the correlation between fitness landscape properties and the choice of operators. In Section 4 we present a genetic programming approach to classify operators on a basis of fitness landscape and a discussion about achieved results and possible avenues for future work. Finally, in Section 5 we give a conclusion.

2. PRELIMINARIES

In this section we present the crossover operators used in the experiments, the employed benchmark functions and fitness landscape properties we consider in the experiments.

2.1 Crossover Operators

We consider only crossover operators for a two-parent case. Naturally, there are many crossover operators and the question is why we chose the following ones. When conducting the experiments we wanted to use as small as possible representative set of operators. Therefore, we chose the operators that are designed with various objectives in mind and that are widely accepted.

If n denotes the dimensionality of the individual, for all crossover operators presented here holds that parents $\beta^{f_1} = \beta_1^{f_1}, \beta_2^{f_1}, ..., \beta_n^{f_1}$ and $\beta^{f_2} = \beta_1^{f_2}, \beta_2^{f_2}, ..., \beta_n^{f_2}$ form a child $\beta^s = \beta_1^s, \beta_2^s, ..., \beta_n^s$, where f_1 and f_2 are the parents' fitness values and s child's fitness value. In some operators, a parameter α is used as a weighting factor whose value is chosen at random over $[0, 1]$, if not specified otherwise.

2.1.1 Arithmetic Crossover

This is probably the most commonly used operator (also called whole arithmetic crossover), which takes the weighted sum of two parental alleles for each gene with the same α [5]:

$$\beta^s = \alpha \cdot \beta^{f_1} + (1 - \alpha) \cdot \beta^{f_2}. \quad (1)$$

2.1.2 Arithmetic Simple Crossover

In this operator a recombination point k is chosen uniformly. Then the first k float values of a random chosen parent are taken and copied into the child. The rest is the arithmetic average of parents 1 and 2 [5]. The arithmetic average is obtained via:

$$\beta^s = \alpha \cdot \beta^{f_1} + (1 - \alpha) \cdot \beta^{f_2}. \quad (2)$$

2.1.3 Arithmetic Single Crossover

In arithmetic single crossover first a random allele k is chosen. At that position the child assumes the arithmetic average of two parents. All other points are copied from a single randomly chosen parent. The expression for the arithmetic average is the same as for the simple arithmetic crossover [5].

2.1.4 Local Crossover

Local crossover is the same as arithmetic crossover, except that the value α is randomly selected for each gene location [4].

$$\beta_i^s = \alpha_i \cdot \beta_i^{f_1} + (1 - \alpha_i) \cdot \beta_i^{f_2}. \quad (3)$$

2.1.5 Average Crossover

Average crossover is whole arithmetic crossover with $\alpha = 0.5$. Two parents β^{f_1} and β^{f_2} generate offspring β^s in the following way [17]:

$$\beta_i^s = (\beta_i^{f_1} + \beta_i^{f_2})/2. \quad (4)$$

2.1.6 BGA Crossover

In BGA crossover two parents β^{f_1} and β^{f_2} generate offspring β^s. Let β^{f_1} be the parent with better fitness. In that case, the offspring has genes calculated in the following way:

$$\beta_i^s = \beta_i^{f_1} \pm rang_i \cdot \gamma \cdot \lambda_i, \quad (5)$$

where λ equals

$$\lambda = \frac{\beta^{f_2} - \beta^{f_1}}{\|\beta^{f_1} - \beta^{f_2}\|}. \quad (6)$$

The sign "-" is selected with 0.9 probability and $rang_i = 0.5 \cdot (\beta^{f_1} - \beta^{f_2})$. Parameter γ equals

$$\gamma = \sum_{k=0}^{15} \alpha_k \cdot 2^{-k}, \quad (7)$$

where $\alpha_i \in 0, 1$ is randomly generated with $p(\alpha_i = 1) = \frac{1}{16}$ [16, 18, 21].

2.1.7 BLX-α Crossover

Blend-α crossover combines two parents β^{f_1} and β^{f_2} to generate offspring β^s by sampling a new value in the range $[min - I \cdot \alpha, max + I \cdot \alpha]$. Here, min and max equal minimum and maximum values respectively for interval $(\beta_i^{f_1}, \beta_i^{f_2})$. I equals $max - min$ [7, 12]. For the α we use value of 0.5.

2.1.8 BLX-α-β Crossover

Blend-α-β operator creates a new offspring by selecting a random value from the interval between the two alleles of the parent solutions β^{f_1} and β^{f_2}. The interval is increased in direction of the solution with better fitness by the factor α, and into the direction of the solution with worse fitness by the factor β [6, 12]. The value for α is 0.75, and for β

0.25. In the case that $\beta_i^{f_1} \leq \beta_i^{f_2}$, random value is sampled from the interval:

$$[\beta_i^{f_1} - I \cdot \alpha, \beta_i^{f_2} + I \cdot \beta], \tag{8}$$

otherwise is sampled from:

$$[\beta_i^{f_2} - I \cdot \beta, \beta_i^{f_1} + I \cdot \alpha]. \tag{9}$$

2.1.9 Discrete Crossover

In discrete crossover an offspring β^s is created from parents β^{f_1} and β^{f_2}, where the allele value for each gene i is given by $\beta_i^s = \beta_i^{f_1}$ or $\beta_i^s = \beta_i^{f_2}$ with equal likelihood [25].

2.1.10 Flat Crossover

Flat crossover generates offspring whose genes are sampled randomly from the interval $min(\beta_i^{f_1}, \beta_i^{f_2}), max(\beta_i^{f_1}, \beta_i^{f_2})$ [18, 20]. This crossover is the same as BLX-α crossover when $\alpha = 0$.

2.1.11 Heuristic Crossover

In heuristic crossover (Wright's heuristic crossover), given two parents β^{f_1} and β^{f_2}, assume that the first parent (β^{f_1}) has a smaller value on current allele [12, 30]. Then the offspring β^s is created as

$$\beta_i^s = \alpha \cdot (\beta_i^{f_2} - \beta_i^{f_1}) + \beta_i^{f_1}, \tag{10}$$

where α is chosen uniformly at random from interval $[0, 1]$.

2.1.12 One Point Crossover

In one point crossover the descendant is created in the same way as in the one point crossover for a binary coded GA. First, choose a random point k. Two parents β^{f_1} and β^{f_2} generate offspring β^{s_1} and β^{s_2}. The first offspring copies allele values from the first parent up to the point k, and from the second parent from that point on. For the second offspring the procedure is analogous [15].

2.1.13 Simulated Binary Crossover

Simulated binary crossover is devised to simulate the effect of one point binary crossover [2, 18]. Two parents β^{f_1} and β^{f_2} generate offspring β^{s_1} and β^{s_2} in the following way:

$$\beta_i^{s_1} = \frac{1}{2}\left[(1 + B_k)\,\beta_i^{f_1} + (1 - B_k)\,\beta_i^{f_2}\right] \tag{11}$$

and

$$\beta_i^{s_2} = \frac{1}{2}\left[(1 - B_k)\,\beta_i^{f_1} + (1 + B_k)\,\beta_i^{f_2}\right], \tag{12}$$

where $B_k \geq 0$ is obtained from the random number source having density

$$B_k(u) = \begin{cases} (2 \cdot u)^{\frac{1}{\eta+1}} & \text{if } u \leq \frac{1}{2} \\ \left(\frac{1}{2 \cdot (1-u)}\right)^{\frac{1}{\eta+1}} & \text{if } u > \frac{1}{2} \end{cases}. \tag{13}$$

Distribution index η is any non-negative real number [3].

2.2 Benchmark Functions

The test problems originate from the COCO platform (COmparing Continuous Optimisers) [10], which has been extensively used in the last few years. Of the functions available we use 24 noise-free real-parameter single-objective problems. The dimensionality D of all the functions is set to 30, which matches a medium-sized optimization problem. The test functions are implemented so the global minimum has the value of zero. The problems can be differentiated on the basis of their modality, where functions 1, 2, 5, 6, 7, 10-14 are unimodal and 3, 4, 8, 9, 15-24 are multimodal.

2.3 Fitness Landscape Properties

Here we present only the properties we used in the correlating fitness landscape and crossover operators. Naturally, there are other properties we could have chosen and indeed adding more properties will be a future goal. Due to the lack of space we do not give discussion about fitness landscape but rather refer to the literature. Likewise, for explanations of the influence of some property to the landscape shape refer to [19, 23]. Fitness landscape can be defined as a set of two functions f and d that define the fitness value and the distance between encoded solutions in the landscape [19].

The diameter of the population P is the maximum distance between the elements of the population [23]:

$$d(P) = max_{s,t \in P} dist(s, t), \tag{14}$$

where $dist(s, t)$ represents L_1 distance [28].

Average distance of the population P is defined as [23]

$$dmm(P) = \frac{\sum_{s \in P} \sum_{t \in P, t \neq s} dist(s, t)}{|P| \cdot (|P| - 1)}. \tag{15}$$

Indicator δ_{dmm} represents the variation of the average distance of the initial population and population after local search algorithm [23]:

$$\delta_{dmm} = \frac{(dmm(U) - dmm(O))}{dmm(U)}, \tag{16}$$

where U is the initial population, and O is population after local search.

Amplitude $Amp(P)$ represents distribution of solutions in objective space, i.e. the difference between the best and worst solution [23]:

$$Amp(P) = \frac{|P| \cdot (max_{s \in P} f(s) - min_{s \in P} f(s))}{\sum_{s \in P} f(s)}. \tag{17}$$

Relative variation of the amplitude Δ_{Amp} of initial and final population [23] is defined as

$$\Delta_{Amp} = \frac{Amp(U) - Amp(O)}{Amp(U)}. \tag{18}$$

The average length of walk $Lmm(P)$ represents the average number of steps needed for the convergence of a solution [23]:

$$Lmm(P) = \frac{\sum_{p \in P} l(p)}{|P|}. \tag{19}$$

Autocorrelation function measures the correlation of the solutions that have distance d [27]:

$$\rho(d) = \frac{\sum_{s,t \in S \times S, dist(s,t) = d} (f(s) - \bar{f})(f(t) - \bar{f})}{n \cdot \sigma_f^2}. \tag{20}$$

Here, n is the number of pairs with distance d. In our experiments we use distance that equals 1.

Correlation length is defined in the following way [22]:

$$l = \frac{1}{ln(|\rho(1)|)}. \tag{21}$$

Autocorrelation coefficient is a normalized measure of autocorrelation function [1].

$$\xi = \frac{1}{1 - \rho(1)}. \tag{22}$$

There are many other properties one could choose and indeed we experimented with other properties. The previous properties were chosen on a basis of the variance of their values for the benchmark functions; e.g. if a property gives roughly the same value for every test function, then it is not a good choice for further experiments. Nevertheless, additional properties can be easily added to the analysis as input variables in genetic programming (in Section 4).

3. CORRELATING CROSSOVER AND FITNESS LANDSCAPE

In this section we first compare individual operators on the set of all problems. Then we try to discern whether there are statistically significant differences between the operators in regards to the specific functions. Finally, we try to correlate between individual crossover operator efficiency on a given function and fitness landscape properties of that function.

3.1 Experimental Setup

In all the experiments with crossover operators, the same selection method and parameter values are used for all the operators. The evolutionary algorithm is a steady-state GA with a tournament selection operator. In every iteration, k individuals are randomly selected for tournament (k is set to 3). The worst of k individuals is identified and replaced with a new individual. The new individual is obtained by crossing the two parents remaining from the tournament. This selection scheme was applied in part because it avoids the use of crossover probability, giving all the operators a fair treatment. Additionally, it has produced the best overall results in different optimization domains in our previous experience.

Mutation is also applied to the new individual with a given mutation rate (0.3 in our experiments). The mutation operator is a simple random change of a random solution variable within the given bounds. We are aware that the influence of mutation may be even more pronounced than with the selection mechanism and crossover. We believe that the differences between crossover operators, if they are significant, should hold even with other mutation operators. On the other hand, conducting experiments without mutation is not in accordance with realistic application conditions.

The population size was set to 100, and the termination criteria is defined as reaching 10^6 fitness evaluations (some operators require an additional evaluation that was accounted for). This number of evaluations allows the algorithm a stable convergence for this problem size (30 variables), which should be a requirement for operator performance comparison. Each combination of benchmark function and operator was tested in 50 runs, and the average (mean) error of the 50 best solutions is recorded.

3.2 Crossover Operator Performance

The obtained results for each operator and each problem are aggregated in Tables 5 and 6. The tables show *normalized error*, where the worst performing operator has the value of 1, and the optimal solution is denoted with zero.

Table 1: Average rankings of the operators (Friedman) and average normalized error

Algorithm	Ranking	$\bar{e}_N$
Arithmetic	6.2917	0.42
Arithmetic Simple	7.0417	0.45
Arithmetic Single	8.3333	0.58
Average	6.875	0.42
BGA	5.625	0.44
BLX-α	8.3333	0.71
BLX-α-β	7.8333	0.69
Discrete	6.625	0.45
Flat	8.4583	0.71
Heuristic	7.625	0.68
Local	5.9167	0.40
One point	7.2917	0.53
SBX	4.75	0.39

From these results, we are interested in discovering significant differences between the operators, which are regarded as different optimization algorithms. Since the resulting values are not normally distributed (the analysis is omitted), we rely on nonparametric methods to reveal if there are significant differences. This is achieved with the Friedman test, which calculates the *average rank* of each algorithm and determines the corresponding P-value. The Friedman results are given in Table 1, and the P-value is equal to 0.016 which means there are significant differences with level of significance $\alpha = 0.05$. The last column in the previous Table gives the *normalized average error* ($\bar{e}_N$), which is defined as the mean normalized error over all the problems.

3.3 Landscape Properties and Crossover Efficiency

The described landscape properties of benchmark functions are obtained with the following: a random population of 1000 individuals is generated within the defined boundaries ($[-50, 50]$ for each variable) and the same initial population is used on every function. A local search algorithm was executed in parallel for every starting point in the population; the algorithm used is based on Hooke-Jeeves directed pattern search. The local search algorithm is run for every starting point until convergence, which was determined with precision of 10^{-6}. After the convergence, the final population is recorded and fitness landscape properties are calculated for each function. The obtained values are given in Table 2.

Just by visually inspecting these values, it is not easy to reach a conclusion whether one can choose the crossover operator on a basis of those data. For instance, it seems there is a correlation between data for functions 2, 6 and 19. In functions 2 and 6 the properties have similar values and BGA crossover performs good, while Flat, BLX-α and BLX-α-β perform poorly. However, for function 19 BGA operator performs poor and Flat, BLX-α and BLX-α-β perform good. For the function 19 we see that fitness landscape properties are consistently different than for functions 2 and 6. We believe that inspecting the properties only visually it is hard to see correlations if they are not obvious, and especially to quantify the impact of those differences.

Table 2: Fitness landscape properties

Problem	d(P)	dmm(P)	Δ_{dmm}	Amp(p)	Δ_{Amp}	Lmm(P)	$\rho(1)$	l	ξ
1	0.000016	0.000003	1	1.618034	-0.034137	64.811	-0.001	0.144765	0.999001
2	0.000009	0.000002	1	2.169216	-0.386414	76.649	-0.001	0.144765	0.999001
3	232.181431	60.475268	0.878971	2.209141	-0.411932	57.769	0.5094	1.482532	2.03832
4	122.409501	32.98095	0.933995	2.608001	-0.666856	59.466	0.519048	1.524952	2.079211
5	650.35941	181.643388	0.636478	1	0.360868	32.475	0.999	999.499917	1000
6	140.178412	6.770292	0.986451	37.165919	-22.753912	2035.275	0.092841	0.420721	1.102342
7	713.30565	93.711719	0.812455	8.515323	-4.442412	160.959	0.313675	0.862516	1.457035
8	8.729045	0.568895	0.998861	8.275631	-4.289217	875.903	0.564717	1.749994	2.297356
9	295.068994	7.855163	0.984279	41.967988	-25.823066	1594.981	0.080365	0.39664	1.087388
10	22.557226	1.041349	0.997916	138.077539	-87.249712	3292.147	0.015926	0.241559	1.016184
11	20.28115	3.482909	0.99303	9.276794	-4.929092	5779.649	0.17005	0.564441	1.204892
12	520.334509	8.894347	0.9822	362.252363	-230.526916	2268.188	0.005114	0.189547	1.00514
13	27.766903	2.906121	0.994184	7.946438	-4.07882	162.833	0.478247	1.355699	1.916617
14	0.493758	0.113324	0.999773	2.49133	-0.592288	632.568	-0.001	0.144765	0.999001
15	391.415693	85.361853	0.829166	2.517099	-0.608758	144.773	0.572866	1.794999	2.341186
16	489.009861	141.624098	0.716568	3.020404	-0.930435	77.137	0.492803	1.413134	1.971619
17	1997.479698	337.798776	0.323965	748.42151	-477.339803	10952.256	0.001762	0.15769	1.001765
18	2175.891213	408.295428	0.18288	853.344799	-544.399588	12795.219	0.001371	0.151686	1.001372
19	877.58457	93.438758	0.813001	3.272264	-1.091407	140.494	0.415787	1.139494	1.711704
20	88.67771	22.475023	0.955021	1.434853	0.08294	218.525	0.000047	0.100428	1.000047
21	158.902852	48.15173	0.903634	2.952137	-0.886804	82.297	0.841152	5.780901	6.29531
22	132.732042	44.09263	0.911758	2.838727	-0.81432	80.208	0.715527	2.987425	3.515267
23	345.344015	90.238947	0.819405	3.86982	-1.473324	86.339	0.424284	1.16638	1.736966
24	159.542561	41.711814	0.916522	1.919647	-0.226907	94.761	0.54411	1.643105	2.193511

4. INFERRING PERFORMANCE WITH A GP-BASED CLASSIFIER

In this section we try to address the following questions: firstly, are the derived fitness landscape properties sufficient to distinguish between suitable and inefficient crossover operators for a specific function? Secondly, is it possible to predict crossover performance based on landscape features in a general case?

4.1 Crossover Operator Classes

To answer the first question, the crossover operators in this work are manually classified into three classes for each function: "good", "average" and "bad" crossovers. The classification is entirely subjective and is based on average operator error (Tables 5 and 6). The main goal here is to discern between the operators, rather than to give a formally justified classification.

The selection is made with the following guidelines: operators belong to different groups if their performances differ by at least a factor of two; if there are no three discernible groups, the "good" and "average" classes are populated first. For instance, for the second function in the benchmark, the division is made as follows: the BGA crossover is the only one considered "good", arithmetic, arithmetic simple, average, local and SBX are considered as "average", while the rest are classified as "bad". On the other hand, for functions 5, 17 and 22 all operators are considered "good" because there is no (significant) difference in their average error.

4.2 A Simple GP-based Classifier

Based on the described layout and available fitness landscape properties, we define a simple classifier based on symbolic regression (more advanced techniques are being investigated). A classifier of the same form is constructed independently for *each* crossover operator, so the final result is a family of classifiers where each is trained and tested separately. The GP classifier uses all landscape properties as terminals and the arithmetic operators (+, -, *, /) to build a tree that produces a value for a given function. Additionally, random constants (ERCs) in $[-1, 1]$ are added to the terminal set. The absolute tree output value is used to determine the operator class: value in $[0, 1]$ corresponds to "good", $[1, 10]$ to "average" and greater values to "bad".

The classifier is trained to minimize the penalty (not entirely equivalent to classification error), which is accumulated for every function. If the operator is classified correctly, the penalty is zero; if the resulting class is different, the penalty is 1 for adjacent classes (e.g. good operator classified as "average") and 3 for opposite classes (e.g. a bad operator classified as "good"). The GP parameters are of secondary importance, but we used a population of 2000 and a stopping criteria of 50 generations without improvement.

4.3 Classification Results

When all the test cases (24 functions) are used as a learning set, the GP is able to produce classifiers for every operator that succeed in obtaining a correct class for every function (i.e. the total penalty is zero). This provides the answer to the first question of whether the landscape properties are correlated with operator performance. A detailed analysis of relevant properties for each operator was not conducted at this stage, but it is to be expected that not every property is equally important for every operator.

The answer to the second question remains a greater challenge. Since this is a preliminary investigation, no specific measures are undertaken to increase the generalization ability of the employed classifier - both the learning model, termination condition and additional test cases are devoted to future research. In this work a simple 4-fold cross-validation is performed on the available set of functions, so that 18

Table 3: Average rankings of the algorithms (Friedman)

Algorithm	Ranking
Arithmetic	7.0208
Arithmetic Simple	7.875
Arithmetic Single	9.2083
Average	7.7708
BGA	6.1458
BLX-α	9.0208
BLX-α-β	8.6042
Discrete	7.375
Flat	9.3125
Heuristic	8.2917
Local	6.6458
One point	8.0417
SBX	5.2708
Learned crx	4.4167

Table 4: Post-hoc analysis (control operator: learned crossover)

i	algorithm	unadjusted p	$p_{Hochberg}$
1	Flat	0.00005	0.000654
2	Arithmetic Single	0.000073	0.00087
3	BLX-α	0.000138	0.001513
4	BLX-α-β	0.000525	0.005252
5	Heuristic	0.001333	0.011996
6	One point	0.002684	0.021472
7	Arithmetic Simple	0.004186	0.029304
8	Average	0.005478	0.032866
9	Discrete	0.014296	0.071481
10	Arithmetic	0.031048	0.124193
11	Local	0.064903	0.194709
12	BGA	0.152176	0.304353
13	SBX	0.479369	0.479369

functions are used for training and the remaining 6 for evaluation in each fold.

For any number of test cases (functions), the maximum penalty would equal to 3 times the number of functions, which corresponds to a 100% normalized penalty. In the cross-validation, penalties on the *evaluation* set on unseen functions for every operator and every fold are recorded. The average penalty of the evaluation set is 17%, which for individual operators varies from 12% to 22%. The achieved values are not spectacular, but we believe that a carefully designed classification process can enhance the quality of these initial results.

4.4 Constructing an Efficient Operator

To test the efficiency of the classifiers, we constructed an artificial algorithm that uses the "recommended" operator for each test function. This method may be considered a hyperheuristic, where a machine learning technique is used to construct the suitable optimization algorithm. For each function, only the operators that classify to "good" are considered; out of these, the operator with the closest to true average rank is selected (the rank is previously available from experimental results in Table 1). For instance, if three operators are classified as "good" for a given function, the one with the true rank closest to average among those three is selected.

The efficiency of such *learned crossover* is compared with individual crossover operators, and the results in terms of average rank are shown in Table 3. The normalized average error ($\bar{e}_N$) obtained for this operator amounts to 0.33, which is the best among the individual crossover operators (Tables 5 and 6).

Based on the Friedman test results in Table 3, a post-hoc statistical analysis is performed to determine where the significant differences are. The analysis is performed by taking the operator with the best rank (which is the learned crossover) as a *control operator* and comparing it with the other operators. The Table 4 presents the results obtained with the Hochberg test; for the level of significance $\alpha = 0.05$, we can conclude that the learned crossover operator is significantly better than the first eight operators from the table (with p-value less than α). For the remaining operators, a statistically significant difference cannot be proven, but the learned operator still provides the best normalized average error.

5. CONCLUSIONS

This paper describes how fitness landscape properties can be used to assess the crossover operator efficiency. The results show that the differences between the operators can be correlated with appropriate landscape properties. A preliminary investigation is performed which shows the potential of fitness landscape properties for operator classification in optimization.

Several issues need to be further investigated in this context. A more diverse set of classifiers (such as decision trees based on information gain) will be employed to enhance the learning and generalization. Also, additional test functions should be investigated, as well as their variations (e.g. rotations, translations, different number of variables).

Various local search algorithms are available for fitness landscape determination that should be investigated. This is especially important in terms of their time complexity, since a very large number of evaluations may be needed for convergence, which can negate the performance gain in a suitable crossover choice.

6. REFERENCES

[1] E. Angel and V. Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science*, 191:229–243, 1998.

[2] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex System*, 9:115–148, 1994.

[3] K. Deb, S. Karthik, and T. Okabe. An introduction to genetic algorithms. In *Sadhana*, pages 293–315. John Wiley & Sons, Inc, 1999.

[4] D. Dumitrescu, B. Lazzerini, L. C. Jain, and A. Dumitrescu. *Evolutionary Computation*. CRC Press, Florida, USA, 2000.

[5] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin Heidelberg New York, USA, 2003.

[6] L. J. Eshelman, K. E. Mathias, and J. D. Schaffer. Crossover operator biases: Exploiting the population distribution. In *ICGA*, pages 354–361, 1997.

[7] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In L. D. Whitley, editor, *FOGA*, pages 187–202. Morgan Kaufmann, 1992.

[8] J. C. Fodor, R. Klempous, and C. P. S. Araujo, editors. *Recent Advances in Intelligent Engineering Systems*, volume 378 of *Studies in Computational Intelligence*. Springer, 2012.

[9] T. D. Gwiazda. *Genetic Algorithms Reference*. Tomasz Gwiazda, 2006.

[10] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

Table 5: Normalized error of crossover operators (functions 1-12)

Operator	1	2	3	4	5	6	7	8	9	10	11	12
Arithmetic	0.0000	0.0290	0.0003	0.0054	0.0000	0.1787	0.4661	0.0595	0.1028	0.1182	0.0994	0.0001
Arithmetic Simple	0.0002	0.0682	0.0011	0.0061	0.0000	0.5278	0.4996	0.0719	0.2203	0.2839	0.2185	0.0003
Arithmetic Single	0.0010	0.3350	0.0107	0.0185	0.0000	1.0000	0.5840	0.1182	0.5585	1.0000	0.9503	1.0000
Average	0.0004	0.1194	0.0042	0.0155	0.0000	0.2040	0.4914	0.1002	0.1063	0.1006	0.0675	0.0001
BGA	0.0000	0.0011	0.0004	0.0011	0.0000	0.0085	0.4515	0.0607	0.2472	0.2852	0.2109	0.0021
BLX-Alpha	0.9717	0.5809	1.0000	0.9977	0.0000	0.3473	0.9833	1.0000	1.0000	0.5270	0.0737	0.0663
BLX-Alpha-Beta	1.0000	0.5437	0.9833	1.0000	0.0000	0.3150	0.9286	0.7809	0.7769	0.5611	0.0785	0.0778
Discrete	0.0021	0.5679	0.0193	0.0240	0.0000	0.4549	0.4197	0.1220	0.3948	0.7851	0.9467	0.0029
Flat	0.9786	0.7980	0.9892	0.9810	0.0000	0.3158	1.0000	0.7865	0.8453	0.5768	0.0780	0.1021
Heuristic	0.9156	0.4816	0.9662	0.9831	0.0000	0.3324	0.9824	0.6980	0.8785	0.4558	0.0753	0.0924
Local	0.0000	0.0411	0.0006	0.0036	0.0000	0.2193	0.4736	0.0794	0.0977	0.0869	0.1038	0.0000
One point	0.0028	1.0000	0.0186	0.0234	0.0000	0.8855	0.4551	0.1348	0.3992	0.9174	1.0000	0.0450
SBX	0.0000	0.0183	0.0000	0.0002	0.0000	0.0322	0.3848	0.0532	0.0715	0.0939	0.1698	0.0000

Table 6: Normalized error of crossover operators (functions 13-24)

Operator	13	14	15	16	17	18	19	20	21	22	23	24
Arithmetic	0.0421	0.0137	0.9631	0.4760	0.9406	0.9298	0.7300	0.4999	0.9658	0.7409	0.7778	1.0000
Arithmetic Simple	0.0549	0.0188	0.9311	0.6340	0.9118	0.9193	0.6315	0.4763	0.8489	0.6901	0.8031	0.8812
Arithmetic Single	0.0755	0.0237	0.8939	0.8006	0.7455	0.7623	0.7941	0.3571	0.6217	0.8869	0.6860	0.6483
Average	0.0712	0.0215	0.9535	0.3941	1.0000	0.9492	0.6234	0.4928	0.8744	0.8337	0.6431	0.9218
BGA	0.0271	0.0037	0.8493	1.0000	0.7726	0.7191	1.0000	0.3922	0.9356	0.7610	1.0000	0.8656
BLX-Alpha	0.9905	0.8047	0.5726	0.2982	0.8222	0.7628	0.4735	0.9062	0.8657	0.9249	0.4530	0.5071
BLX-Alpha-Beta	0.9938	1.0000	0.5470	0.2959	0.8540	0.7427	0.4789	0.8845	0.8267	0.9349	0.4363	0.5004
Discrete	0.0563	0.0189	0.6670	0.5874	0.6839	0.5918	0.6409	0.3125	0.5065	0.8468	0.6215	0.6030
Flat	0.9621	0.8230	0.5858	0.3047	0.8700	0.7543	0.5006	1.0000	0.8098	1.0000	0.4391	0.4945
Heuristic	1.0000	0.7828	0.5812	0.2973	0.8368	0.7038	0.4738	0.9014	0.9029	0.9170	0.4473	0.4981
Local	0.0393	0.0137	1.0000	0.3466	0.8813	1.0000	0.6469	0.5310	0.8176	0.8916	0.5649	0.7811
One point	0.0555	0.0201	0.6345	0.7267	0.6432	0.6549	0.9526	0.3101	0.7536	0.7615	0.6421	0.5908
SBX	0.0261	0.0085	0.8301	0.5889	0.8882	0.9155	0.5217	0.4993	1.0000	0.7979	0.7630	0.7468

[11] F. Herrera, M. Lozano, and A. Sánchez. Hybrid crossover operators for real-coded genetic algorithms: an experimental study. *Soft Computing*, 9(4):280–298, 2005.

[12] F. Herrera, M. Lozano, and A. M. Sánchez. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*, 18:309–338, 2003.

[13] E. Masehian, H. Akbaripour, and N. Mohabbati-Kalejahi. Landscape analysis and efficient metaheuristics for solving the n-queens problem. *Comp. Opt. and Appl.*, 56(3):735–764, 2013.

[14] G. Merkuryeva and V. Bolshakovs. Benchmark fitness landscape analysis. *International Journal of Simulation*, 12(2), 2011.

[15] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, UK, 1996.

[16] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evol. Comput.*, 1(1):25–49, Mar. 1993.

[17] T. Nomura. An analysis on crossovers for real number chromosomes in an infinite population size. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence - Volume 2*, IJCAI'97, pages 936–941, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[18] D. Ortiz-Boyer, C. Hervas-Martinez, and N. Garcia-Pedrajas. Improving crossover operators for real-coded genetic algorithms using virtual parents. *Journal of Heuristics*, (13):265–314, 2007.

[19] E. Pitzer and M. Affenzeller. A comprehensive survey on fitness landscape analysis. In J. Fodor, R. Klempous, and C. Suárez Araujo, editors, *Recent Advances in Intelligent Engineering Systems*, volume 378 of *Studies in Computational Intelligence*, pages 161–191. Springer Berlin Heidelberg, 2012.

[20] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex System*, 2(5):183–205, 1991.

[21] D. Schlierkamp-voosen and H. Mühlenbein. Strategy adaptation by competing subpopulations. In *Parallel Problem Solving from Nature (PPSN III*, pages 199–208. Springer-Verlag, 1994.

[22] P. F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20(1):1–45, 1996.

[23] E.-G. Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009.

[24] J. Tavares and F. B. C. Pereira. Golomb rulers: A fitness landscape analysis. In *IEEE Congress on Evolutionary Computation*, pages 3695–3701. IEEE, 2008.

[25] H.-M. Voigt, H. Mühlenbein, and D. Cvetkovic. Fuzzy recombination for the breeder genetic algorithm. In *Proc. Sixth Int. Conf. on Genetic Algorithms*, pages 104–111. Morgan Kaufmann Publishers, 1995.

[26] J.-P. Watson. *An Introduction to Fitness Landscape Analysis and Cost Models for Local Search*, volume 146, chapter 20, pages 599–623. Springer US, Boston, MA, 2010.

[27] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990.

[28] S. Willard. *General Topology*. Dover Publications, 2004.

[29] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[30] A. H. Wright. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*, pages 205–218. Morgan Kaufmann, 1991.