Ensembled Crossover based Evolutionary Algorithm for Single and Multi-objective Optimization

COIN Report Number 2021010

Shreya Sharma
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India
cs5170493@iitd.ac.in

Julian Blank
Department of Computer Science and Engineering
Michigan State University
East Lansing, USA
blankjul@egr.msu.edu

Kalyanmoy Deb, *Fellow, IEEE*
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, USA
kdeb@egr.msu.edu

Bijaya Ketan Panigrahi
Department of Electrical Engineering
Indian Institute of Technology Delhi
New Delhi, India
bkpanigrahi@ee.iitd.ac.in

***Abstract*—A unique way evolutionary algorithms (EAs) are different from other search and optimization methods is their recombination operator. For real-parameter problems, it takes two or more high-performing population members and blends them to create one or more new solutions. Many real-parameter recombination operators have been proposed in the literature. Each operator involves at least a parameter that controls the extent of exploration (diversity) of the generated offspring population. It has been observed that different recombination operators and specific parameters produce the best performance for different problems. This fact imposes the user to use different operator and parameter combinations for every new problem. While an automated algorithm configuration method can be applied to find the best combination, in this paper, we propose an Ensembled Crossover based Evolutionary Algorithm (EnXEA), which considers a number of recombination operators simultaneously. Their parameter values and applies them with a probability updated adaptively in proportion to their success in creating better offspring solutions. Results on single-objective and multi-objective, constrained, and unconstrained problems indicate that EnXEA's performance is close to the best individual recombination operation for each problem. This alleviates the use of expensive parameter tuning either adaptively or manually for solving a new problem.**

***Index Terms*—Crossover, Recombination, Ensemble-based algorithm, Evolutionary algorithm**

I. Introduction

The modular aspect of several operators, such as selection, crossover or recombination, mutation, and survival selection, makes evolutionary algorithms (EAs) relatively easy to experiment with in solving various types of problems. Since their first suggestions, various versions of each operator have been proposed. Although such studies are essential, many versions of an operator make it confusing and challenging to choose the single best set of operators for a specific problem. There have been two approaches followed in the literature. Most studies choose the evolutionary operators by intuitive means of understanding the problem and selecting an operator that is most suited according to the vast EA literature for a similar application or by adopting the past study operators. An alternative to this approach is to use an ensemble-based framework, in which several versions of each operator are considered as possible alternatives and the ensemble method dynamically finds the most appropriate operators for a specific problem. This paper uses the latter approach, but focuses on the crossover operator alone to solve real-parameter-based single and multi-objective optimization problems.

The recombination operator is one of the operators that makes EAs distinguishable from other optimization algorithms and is considered one of the key operators. It takes two or more above-average population members and creates one or more offspring for the next generation. The operator is expected to combine good aspects from multiple better parents into a single offspring solution, thus making it one of the key contributors of the so-called "implicit parallelism" aspect of EAs. In real-parameter problems, most suggested recombination operators *blend* two or more variable vectors into a new vector either by variable-wise blending or vector-wise operations. Most successful implementations of real-parameter recombination operators have a self-adaptive property due to a key aspect in their working principle – the difference between created offspring and parents creating them is in proportion to the diversity of the parental solutions. Thus, when parents are spread well across the entire search space, offspring solutions are also created across the search space, thereby allowing a good global search early on in the optimization process. However, when the parent population converges to

near potential good regions of the search space, the same recombination operator helps to focus the search by creating offspring solutions close to the parents. Simulated binary crossover (SBX) [1], blend crossover (BLX) [2], differential evolution (DE) [3] operators are some examples for self-adaptive recombination operators.

Even though these different self-adaptive recombination operators use a similar principle, their effectiveness differs from problem to problem. Thus, a user must investigate which operators are more suitable for which problems. Finding a good performing recombination operator can become cumbersome and may take a lot of computational burden. In this paper, we propose an ensembled crossover operator that chooses the most appropriate one generation-wise from a set of recombination operators for the specific problem to alleviate the computational burden.

In the rest of the paper, we outline some existing studies on ensemble-based EAs in Section II, followed by a motivational aspect of this study in Section III. Then, in Section IV, we provide a detailed description of the methodology used in this study. Results are then presented in Section V. Finally, conclusions and future extensions of this study are provided in Section VI.

II. Related Work

Ensemble-based EAs can be used to choose a) the most-suitable algorithm, b) the most appropriate operator for selection, recombination, or mutation, or c) the most appropriate parameter for a specific operator. We discuss the existing EA studies in the following.

A. Ensemble in Algorithm Selection

AMALGAM [4] is a hybrid optimization framework that employs four sub-algorithms simultaneously, including NSGA-II, Adaptive Metropolis Search, Particle Swarm Optimization, and Differential Evolution. It is designed to overcome the drawbacks of using a single algorithm. The strategies of global information sharing and genetically adaptive offspring creation are implemented in the process of population evolution. Each sub-algorithm is allowed to produce a specific number of offsprings based on the previous generation's survival history. The pool of current best solutions is shared among sub-algorithms for reproduction.

Based on the steady-state structure of ε-MOEA, Borg [5] incorporates more advanced features into a unified framework, including ε-dominance, ε-progress (a measure of convergence speed), randomized restart, and auto-adaptive multi-operator recombination (similar to AMALGAM). The advantages of Borg are threefold: (1) the usage of ε-box dominance archive, which maintains convergence and diversity concurrently throughout the search; (2) the combination of time continuation, adaptive population sizing, and two types of randomized restarts (ε-progress triggered restart and population-to-archive ratio triggered restart) which guides the algorithm towards the global optimum; (3) the simultaneous employment of multiple recombination operators to enhance the performance on a wide assortment of problem domains. Also, the adoption of the steady-state, elitist model of ε-MOEA makes it easily extendable to make use of parallel computing.

B. Ensemble in Operator Selection

Ensemble-based methods [6] allow benefiting from different recombination operators with varying parameter values whenever they are effective during different search processes. Recently, Adaptive EP (AEP) using Gaussian (ACEP) and Cauchy (AFEP) mutations have been proposed. In the AEP, the strategy parameter values are adapted based on the previous generations' search performances. The ensemble's performance is compared with a mixed mutation strategy, integrating several mutation operators into a single algorithm and with the AEP with a single mutation operator. Results indicate superior performance of the ensemble compared to the single mutation-based and mixed mutation algorithm.

C. Ensemble in Parameter Selection

While an ensemble-based parameter selection can be performed, finding the most suitable algorithm configuration has a similar purpose [7]–[9]. Some widely used methods and frameworks for optimizing algorithm configurations are irace [10], Optuna [11], and Hyperopt [12]. These methods use a bi-level optimization algorithm in which the upper-level optimization task searches for suitable hyper-parameter values (such as operator probabilities and/or population size), and the lower-level optimizes the specific problem set to evaluate the chosen hyper-parameter values. While these methods can be used for parameter selection, the concept can also be used for algorithm or operator selection.

To the best of our knowledge, no study has solely focused on an ensemble-based crossover operator, which shall be the focus of this paper. Moreover, we investigate why one specific algorithm/operator/parameter works well on certain problem types, which has often been neglected in the literature.

III. Motivation for the Study

Over the years, EA researchers have proposed several alternative recombination techniques for real-parameter EAs. Some such alternatives are simulated binary crossover, biased crossover, differential evolution crossover, uniform crossover operators, etc.

Different combinations of parameter values may be suitable for different problems. To demonstrate, we consider five single-objective test problems from the literature and apply six different recombination operator-parameter combinations. Simulated binary crossover (SBX) [1] is used with two distribution index parameters: (i) $\eta_c = 20$ (SBX(20)) and (ii) $\eta_c = 50$ (SBX(50)), (iii) biased crossover (BX(0.5)) in which every variable is chosen randomly with a probability 0.5 from each parent mimicking the uniform crossover operator of binary-coded GAs, (iv) parent-centric crossover (PCX(0.1)) with variance parameters $w_\zeta = w_\eta = 0.1$ [13], (v) blend crossover (BLX(0.5)) is used with $\alpha = 0.5$ [2], (vi) and

differential evolution [3] with $F = 0.3$ and without the variable exchange operator are used.

We present the difference between the obtained and true optimal objective values (Normalized IGD) in Figure 1. It becomes obvious that no crossover operator works the best across all six problems. For instance, while DE works the best for *rosenbrock* problem, it performs the worst for *ackley* and *sphere* problems.

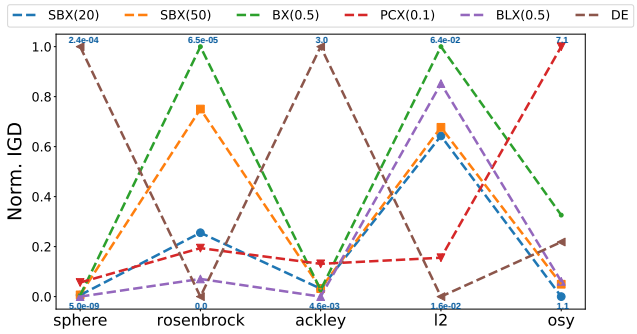


Fig. 1: Effect of various crossover operators on different optimization problems.

Since different operators perform better on different optimization problems, one may not know this crucial information beforehand. Thus, we propose an ensemble-based approach for operator selection, eliminating the need to decide what crossover operator to use or run an automated algorithm configuration approach beforehand. Thus, no trial and error approach is required to find the most suitable recombination operator and its parameter values for a problem.

IV. Problem Methodology

We propose an Ensembled Crossover-based Evolutionary Algorithm (EnXEA), which uses multiple crossover operators at a time. The proposed approach dynamically uses more suitable crossovers by taking their past success rate into account. A dynamic crossover selection addresses the fact that different crossovers are suitable for different kinds of optimization problems. Thus, the proposed crossover ensemble shall be more robust on various optimization problems with different characteristics such as variable dependency or multi-modality.

An outline of EnXEA is shown in pseudo-code in Algorithm 1. Overall, the algorithm follows the typical procedure of an evolutionary algorithm. First, the initial population is created and evaluated (Lines 1 to 2). Afterward, the β-vector reflecting the crossover's success rates is uniformly initialized, making each crossover C_i equally likely to be used during the mating. Then, in each generation of the evolutionary algorithm, the mating is adapted to use not one but multiple crossovers (Line 6 to 11). After having initialized an empty set of offspring $O^{(t)}$ the algorithm selects crossover k based on

Algorithm 1: EnXEA: An Ensembled Crossover based Evolutionary Algorithm

Input: Crossover Operator Set C, Population Size N, Maximum Generations G

```
/* Initialize N pop. members and evaluate */
1  P^(0) ← Initialize()
2  Evaluate(P^(0))
/* Initialize crossovers with equal prob. */
3  foreach k ← 1 to |C| do β_k ← 1/|C|;
/* For each generation until termination */
4  foreach t ← 1 to G do
       /* Create the offsprings through mating */
5      O^(t) = ∅
6      while |O^(t)| ≤ N do
           /* Calculate p_k using β (Eqn. 1 to 6)
              and sample from prob. distribution */
7          k ← Select_Crossover_Index(β)
8          c ← Mating(C_k, P^(t−1))
9          c.id ← k
10         O^(t) ← O^(t) ∪ c
11     end
12     Evaluate(O^(t))
13     P^(t) ← Survival(P^(t−1) ∪ O^(t))
       /* Update the crossover probabilities */
14     foreach k ← 1 to |C| do
15         β_k ← {Σ_{c∈P^(t)\P^(t−1)} 1 | c.id=k} / {Σ_{c∈O^(t)} 1 | c.id=k}
16     end
17 end
```

a probability distribution derived from the β-vector (Line 7). Probability updates used in this study are provided below:

$$p_k^{(1)} = 1/|C|, \quad \text{(uniform)} \quad (1)$$

$$p_k^{(2)} = \begin{cases} 1/|C|, & \text{if } \epsilon < 0.1 \\ \kappa_2 \beta_k, & \text{otherwise} \end{cases} \quad \text{(epsilon)} \quad (2)$$

$$p_k^{(3)} = \kappa_3 \exp(\beta_k), \quad \text{(exponential)} \quad (3)$$

$$p_k^{(4)} = \kappa_4 \beta_k, \quad \text{(linear)} \quad (4)$$

$$p_k^{(5)} = \kappa_5 \beta_k^2, \quad \text{(quadratic)} \quad (5)$$

$$p_k^{(6)} = \begin{cases} 1, & \text{if } \operatorname{argmax}(\beta) = k \\ 0, & \text{otherwise} \end{cases} \quad \text{(maximum)} \quad (6)$$

The constant κ_j $(j = 1, \ldots, 6)$ is fixed such that $\sum_{k=1}^{|C|} p_k^{(j)} = 1$ for all j. The mating using the k-th crossover (C_k) results in the offspring set c, which is then added to the offspring population $O^{(t)}$. The procedure is repeated until the offspring population has reached the required population size. After evaluating the offspring and performing the survival selection, each crossover's success rate stored in the β-vector is updated based on the proportion of survivors to the created

offspring members (Lines 14 to 16). After the probability update, the current generation ends. The algorithm is terminated when the maximum number of allocated generations is reached.

The proposed method extends the traditional genetic algorithm by using not a single but multiple crossovers. Using more than one crossover inevitable raises the question of what pool of crossovers and how frequently each of them should be used, which shall be addressed through an empirical study next.

V. Results

In this section, we present the results of this study. The number of variables, objectives, and constraints for all our experiments are listed in Table I. We have used the well-known multi-objective optimization algorithm NSGA-II [14] for optimizing single and multi-objective problems, and NSGA-III [15], [16] for many-objective problems. We have fixed the population size to 100 for both algorithms across all runs and run it for the number of generations, as shown in the table below. The implementation of the algorithms and evolutionary operators is based on pymoo [17], a framework for multi-objective optimization written in Python.

TABLE I: Optimization problems and their parameters used in this study.

	Problem	#Var.	#Obj.	#Constr.	#Gen.
Single-objective	ackley	10	1	0	100
	sphere	10	1	0	100
	rosenbrock	10	1	0	100
	schwefel	2	1	0	100
	G01	13	1	9	100
	G04	5	1	6	200
Multi-objective	ZDT1	30	2	0	100
	ZDT2	30	2	0	100
	ZDT3	30	2	0	100
	ZDT4	10	2	0	150
	ZDT6	10	2	0	150
	L1	30	2	0	500
	L2	30	2	0	300
	DTLZ1	7	3	0	100
	DTLZ2	10	3	0	100
	DTLZ3	10	3	0	300
	TNK	2	2	2	100
	OSY	6	2	6	200

A. Validating the Ensemble Concept

To investigate if the proposed ensemble method is working or not, we perform a simple experiment with two-objective ZDT1 and ZDT2 problems [18]. We know from their construction that all Pareto-optimal solutions must have $x_i = 0$ for $i \geq 2$. Knowing $x_i \in [0, 1]$, if we introduce a crossover operator in which every variable $i \geq 2$ inherits the smaller value from the parents at index i. Such a customized crossover shall help to converge to the Pareto-front rapidly. We call this operator "SBX_min(5)", as the first variable is still recombined with $\eta_c = 5$. To investigate, we include another crossover operator ("SBX_max(100)"), which assigns the larger of both parents to the child for $i \geq 2$ and SBX with $\eta_c = 100$ is used for the first variable.

For this study, we use the quadratic update rule (Equation 5), in which the probability of selecting a crossover operator is assigned in proportion to the number of survived offspring created by the operator in the next generation. As can be seen from Figures 2 and 3 that the "SBX_min(5)" crossover is selected by our ensemble method more often than any other operator, as our overall Algorithm 1 detects that it creates more successful offspring and thereby increasing its usage with generations is beneficial. Similarly, the "SBX_max(100)" crossover operator is not chosen much due to its creation of worse offspring solutions for both problems, as predicted. Similar behavior is also observed for the ZDT2 problem in Figure 3.

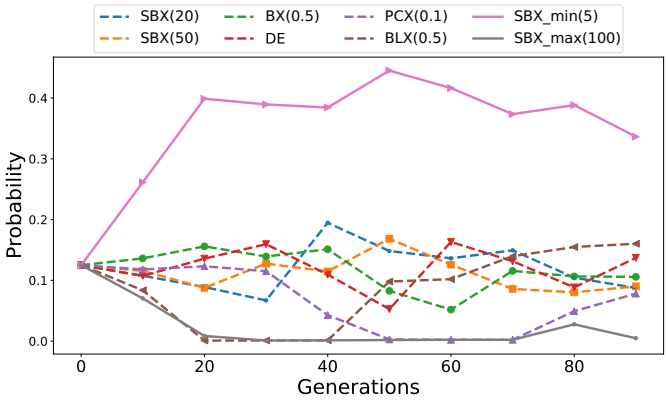


Fig. 2: Probability of choosing a crossover operator for ZDT1.

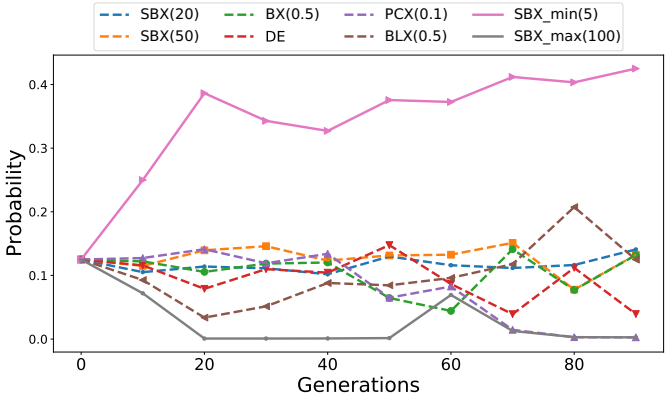


Fig. 3: Probability of choosing a crossover operator for ZDT2.

Having shown that our proposed algorithm selects the right crossover operator for a problem, we now investigate the probability update rule's effect on the overall algorithm's performance in the next subsection.

B. Validating Proposed Probability Update Method

We have ($K = |C|$) different crossover operators. Once the proportion (β_k) of the surviving offspring population created

by a specific crossover operator k in the offspring population is calculated (described in Lines 14 to 16), the probability of using the k-th crossover operator $p_k^{(j)}$ must be assigned using one of different update rules (j-th rule).

Figure 4 shows the performance of EnXEA using six update rules on six multi-objective problems. The ranking of each update rule is also tabulated in Table II. It is clear from the figure and table that the quadratic update rule performs the best. Based on this outcome, we use the quadratic update rule for the rest of the paper.

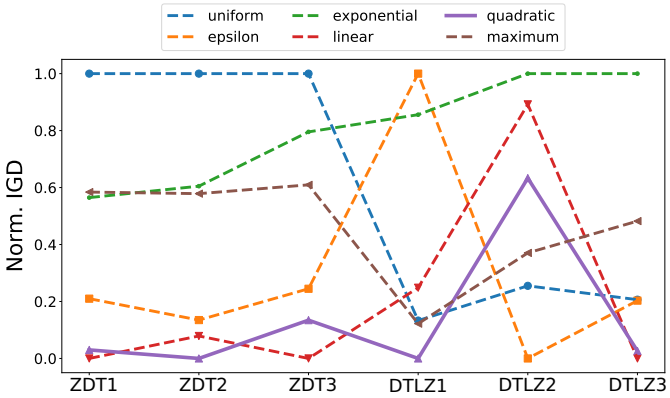


Fig. 4: Performance of EnXEA for different probability update functions.

TABLE II: Performance ranking of EnXEA for different update functions on different problems.

Problem	Uni.	Eps.	Exp.	Lin.	Quad.	Max.
ZDT1	6	3	4	1	2	5
ZDT2	6	3	5	2	1	4
ZDT3	6	3	5	1	2	4
DTLZ1	3	6	5	4	1	2
DTLZ2	2	1	6	5	4	3
DTLZ3	4	3	6	1	2	5
Avg.	4.5	3.2	5.2	2.3	**2.0**	3.8

C. Results on Single-objective Optimization Problems

The normalized IGD values (difference of obtained f^* from known optimal objective value) for six single-objective problems are plotted in Figure 5. The first four problems are unconstrained, while the final two problems (G01 and G04) are constrained problems. It is clear from the figure that our proposed EnXEA has performed better overall on all problems. Figure 6 shows the box-plots of IGD value (difference between obtained and true optimal function values) for 31 runs for each problem. EnXEA makes a better overall performance on all problems than any individual crossover operator.

Figure 7a shows the probability of choosing six crossover operators with generations. The DE operator is chosen least frequently for the sphere problem, while SBX(20), SBX(50), and BX(0.5) are chosen most often during the optimization process. This agrees with the performance of individual crossover operators shown in the box-plot in Figure 6b. For

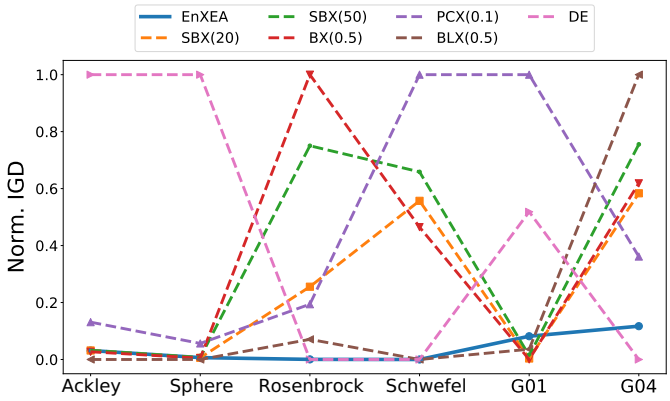
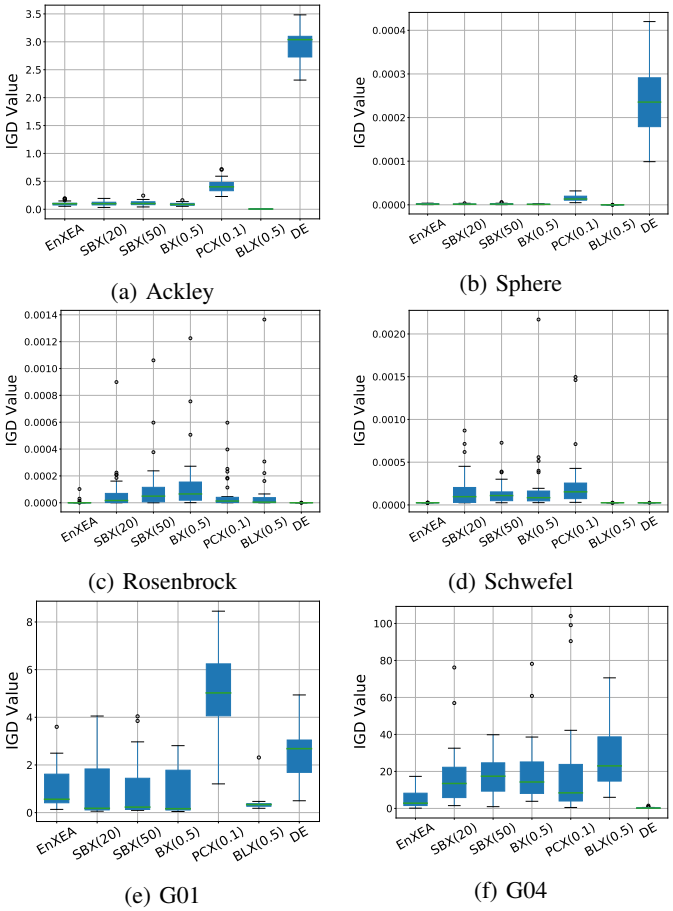


Fig. 5: Performance on single-objective problems indicating superior overall performance by proposed EnXEA.



(a) Ackley (b) Sphere

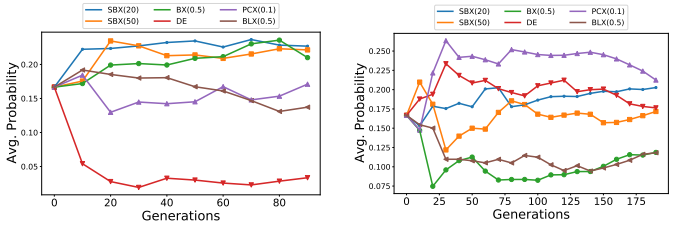
(c) Rosenbrock (d) Schwefel

(e) G01 (f) G04

Fig. 6: Box plots for showing performance of EnXEA compared to other crossover operators for single objective problems.

the constraint G04 problem, BLX(0.5) and BX(0.5) are chosen less frequently, while PCX(0.1), SBX(20), and DE are chosen more frequently.

Table III presents the median IGD value for individual crossover operators and EnXEA for single-objective problems.



(a) Sphere - Unconstrained (b) G04 - Constrained

Fig. 7: Crossover selection probability update plots of EnXEA for single objective problems

The best IGD value is marked in bold, while the others with statistically similar performance (according to the Wilcoxon Rank Sum Test with 95% confidence level) are marked in bold-italics. While DE and BLX(0.5) perform the best for single-objective problems, EnXEA is close to the best performing operator.

TABLE III: Difference in function values from optimal f^* for single-objective problems.

Problem	EnXEA	SBX (20)	SBX (50)	BX (0.5)	DE	PCX (0.1)	BLX (0.5)
Ackley	1e-01	1e-01	1e-01	9e-02	3.0	4e-01	**5e-03**
Sphere	2e-06	1e-06	2e-06	1e-06	2e-04	1e-05	**5e-09**
Schwefel	***3e-05***	1e-04	1e-04	8e-05	***3e-05***	2e-04	**3e-05**
Rosen	3e-08	2e-05	5e-05	7e-05	**0.00**	1e-05	4e-06
G01	6e-01	***2e-01***	***2e-01***	**2e-01**	2.7	5.0	***4e-01***
G04	2.8	13.5	17.4	14.3	**2e-01**	8.4	23.0

D. Results on Multi-objective Optimization Problems

The normalized IGD values for six two-objective problems are plotted in Figure 8. The problems $L1$ and $L2$ are linked-variable problems described below [19]:

Problem L1:

$$\begin{aligned} &\text{Minimize } f_1(\boldsymbol{x}) = (1+g(\boldsymbol{x}))x_1, \\ &\text{Minimize } f_2(\boldsymbol{x}) = (1+g(\boldsymbol{x}))(1-x_1^{0.5}), \\ &\text{Where } g(\boldsymbol{x}) = \sum_{i=2}^{10} |x_i - (0.2+0.6x_1)^2|^{0.6}, \\ &\qquad x_i \in [0,1], \quad \forall i = 1,2,\ldots,10. \end{aligned} \tag{7}$$

Problem L2:

$$\begin{aligned} &\text{Minimize } f_1(\boldsymbol{x}) = (1+g(\boldsymbol{x}))x_1, \\ &\text{Minimize } f_2(\boldsymbol{x}) = (1+g(\boldsymbol{x}))(1-x_1^{0.5}), \\ &\text{Where } g(\boldsymbol{x}) = \sum_{i\in I_1} |x_i - (0.2+0.6\cos(0.5\pi x_1))| \\ &\qquad + \sum_{i\in I_2} |x_i - (0.2+0.6\sin(0.5\pi x_1))|, \\ &\qquad I_1 = \{i | i \text{ is odd and } 2 \le i \le 10\}, \\ &\qquad I_2 = \{i | i \text{ is even and } 2 \le i \le 10\}, \\ &\qquad x_i \in [0,1], \quad \forall i = 1,2,\ldots,10. \end{aligned} \tag{8}$$

Although SBX(20), SBX(50), and BX(0.5) perform the best for ZDT problems, on the two linked problems, the standard SBX or the uniform crossover (BX(0.5)) does not work so well. However, the proposed EnXEA chooses the appropriate operator to produce an excellent performance on all six problems. To demonstrate the effect of multiple runs, we

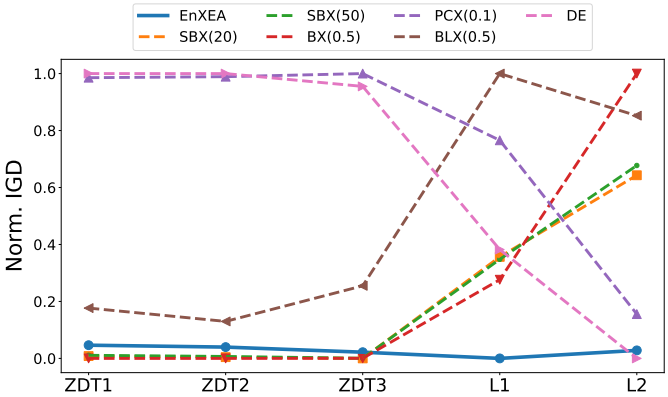


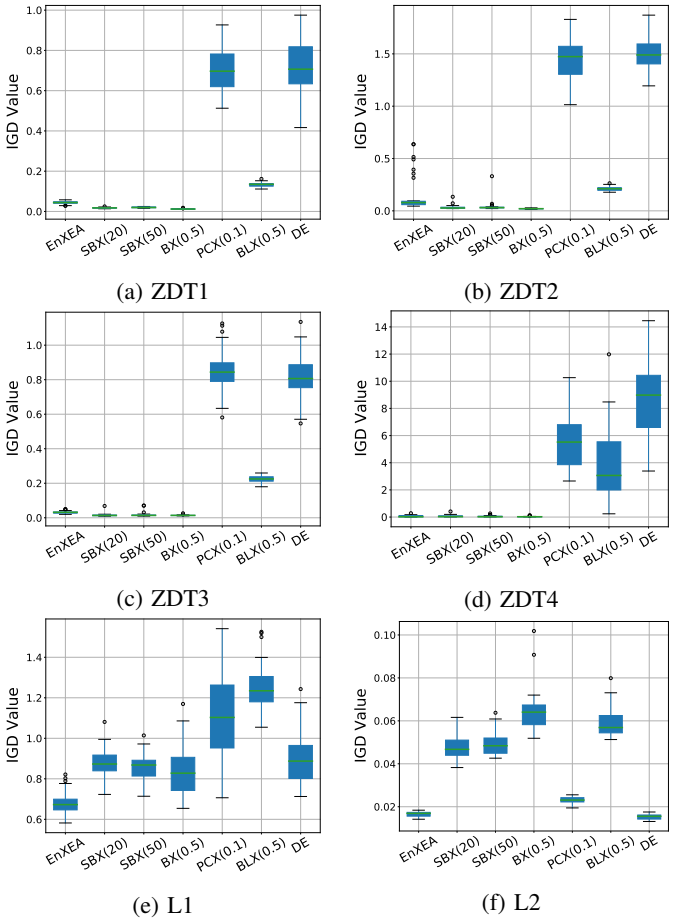
Fig. 8: Performance on ZDT problems, indicating superior overall performance by proposed EnXEA.

present box-plots (from 31 runs) for ZDT1, ZDT2, ZDT3, ZDT4 problems and the linked problems L1 and L2 in Figure 9. It is clear from the sub-figures that SBX and BX operators perform well on ZDT1 and ZDT2 problems due to their variable-independent nature but do not perform well for the linked problems L1 and L2. In these latter problems, DE and PCX(0.1) (partially) perform better. However, since EnXEA can choose any of these crossover operators during the optimization process, it performs reasonably well on all six problems.

Figure 11 shows the probability update of different crossover operators during a specific optimization run of EnXEA on DTLZ1 [20] and TNK [21]. The sub-figures show a differential choice of operators as the run progresses, indicating the working of our proposed probability update procedure. For DTLZ1, SBX operators are chosen more frequently, mainly due to the problem's variable-independent nature. For the constrained TNK problem, PCX(0.1) and SBX(50) are chosen more frequently due to variable interactions needed to stay on a nonlinear Pareto-optimal front.

Figure 12 shows the performance of individual operators and EnXEA on three-objective DTLZ problems and on two constrained problems (TNK [21] and OSY [22]). Similar conclusions about the better performance of EnXEA can be drawn from the figure. Table IV shows the median IGD for all individual operators and EnXEA on two and three-objective problems. EnXEA performs relatively well compared to other operators.

By accumulating the performance on all single and multi-objective problems and noting their algorithm's ranks, we tabulate the average rank of each operator and EnXEA in Table V. The results indicate that EnXEA performs best overall. Interestingly, every individual operator and EnXEA performs

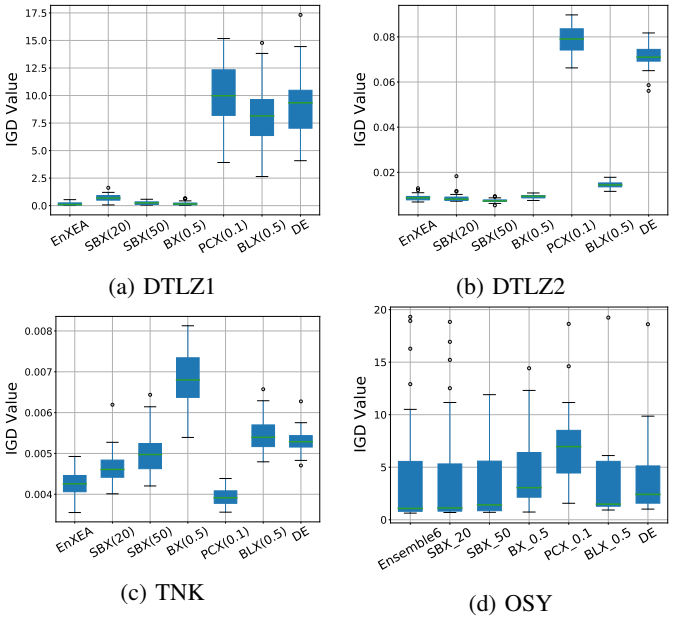


(a) ZDT1 (b) ZDT2

(c) ZDT3 (d) ZDT4

(e) L1 (f) L2

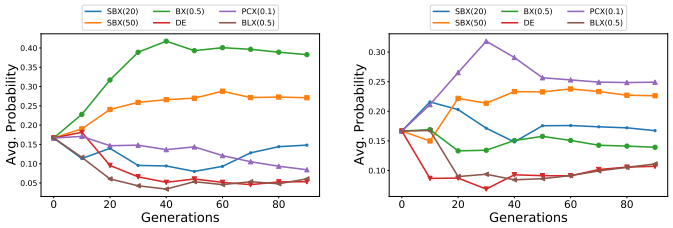
Fig. 9: Box plots for showing performance of EnXEA compared to other crossover operators for ZDT-type problems



(a) DTLZ1 (b) DTLZ2

(c) TNK (d) OSY

Fig. 10: Box plots for showing performance of EnXEA compared to other crossover operators for multi-objective problems



(a) DTLZ1 - Unconstrained (b) TNK - Constrained

Fig. 11: Crossover selection probability update plots of EnXEA for multi-objective problems

TABLE IV: Median IGD values for multi- and many-objective problems.

Problem	EnXEA	SBX (20)	SBX (50)	BX (0.5)	DE	PCX (0.1)	BLX (0.5)
ZDT1	0.044	0.018	0.02	**0.012**	0.707	0.697	0.135
ZDT2	0.079	0.027	0.031	**0.021**	1.490	1.474	0.212
ZDT3	0.032	**0.014**	***0.014***	***0.014***	0.807	0.844	0.225
ZDT4	***0.024***	0.032	***0.026***	**0.024**	8.979	5.521	3.064
L1	**0.673**	0.873	0.868	0.828	0.887	1.103	1.235
L2	0.017	0.047	0.048	0.064	**0.016**	0.023	0.057
DTLZ1	**0.098**	0.69	0.247	***0.141***	9.336	9.980	8.149
DTLZ2	0.009	0.008	**0.007**	0.009	0.071	0.079	0.014
DTLZ3	**0.022**	0.179	0.061	***0.037***	40.31	36.168	21.676
TNK	0.439	0.466	0.503	0.686	0.531	**0.394**	0.547
OSY	**1.078**	***1.121***	***1.414***	3.053	2.415	7.047	1.482

the best on at least one problem. EnXEA and SBX(20) have the least variation in rank. EnXEA and BX(0.5) have the same median (2), but BX(0.5) has a larger variation, including the worst performance in three problems.

VI. Conclusions and Extensions

The crossover operator is the most important and unique in evolutionary algorithms. Thus, it is not surprising that many different crossover operators have been proposed for different problem-solving tasks, including real-parameter optimization. However, it has been observed that one crossover operator works well on specific problems but does not work so well in other problems. In this paper, we have proposed an ensemble-based crossover that chooses different crossover operators using a probability vector derived from operators' success in the past generation. Different probability update rules have been considered, and a quadratic update has been found to perform the best. While other frequency updates have been considered, executing the update at every generation is found to perform the best.

On six single-objective and 11 multi-objective optimization problems, the proposed EnXEA has been found to have the best overall rank on 31 runs compared to an EA with individual crossover operators. EnXEA also has the least mean, median, and variation in rank over 17 problems. A generation-wise plot of the probability vector reveals the selection of crossover operators for different problems. We have observed that the variable-wise crossover operators (SBX, BLX, and BX) perform the best on variable-separable or low-linkage single and multi-objective problems. For linked problems, vector-wise

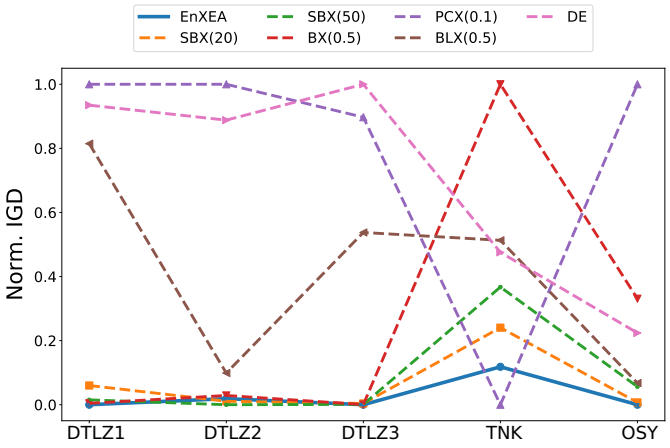


Fig. 12: Performance on three-objective DTLZ and constrained two-objective problems.

TABLE V: Difference in function values from optimal f^* for unconstrained and constrained, single and multi-objective optimization problems.

Problem	EnXEA	SBX (20)	SBX (50)	BX (0.5)	DE	PCX (0.1)	BLX (0.5)
Sphere	4	3	5	2	6	1	7
Rosenbrock	2	5	6	7	4	3	1
Ackley	3	4	5	2	6	1	7
Schwefel	3	5	6	4	7	1	2
G01	5	2	3	1	7	4	6
G04	2	4	6	5	3	7	1
Zdt1	4	2	3	1	6	5	7
Zdt2	4	2	3	1	6	5	7
Zdt3	4	1	3	2	7	5	6
Zdt4	2	4	3	1	6	5	7
L1	1	4	3	2	6	7	5
L2	2	4	5	7	3	6	1
Dtlz1	1	4	3	2	7	5	6
Dtlz2	3	2	1	4	7	5	6
Dtlz3	1	4	3	2	6	5	7
Tnk	2	3	4	7	1	6	5
Osy	1	2	3	6	7	4	5
Min Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Med Rank	**2**	4	3	**2**	6	5	6
Max Rank	**5**	**5**	6	7	7	7	7
Mean Rank	**2.59**	3.25	3.82	3.29	5.59	4.41	5.06

crossover operators (DE and PCX) have performed better.

The study can be extended to many-objective optimization problems, which we plan to execute next. We also plan to compare this study's results with hyper-parameter tuning methods, knowing that generic hyper-parameter tuned values may not be best for individual problems. A bi-level operator-parameter ensemble approach can be tried in which the upper-level ensemble method deals with operators only. Depending on the chosen operators, a lower-level can deal with the operator's hyper-parameter values. Extension of the study to other problems, such as combinatorial problems or mixed-variable problems, would be interesting. Moreover, an extension may consider not only the crossover and but also the mutation operator. This more comprehensive approach introduces another layer of complexity because the mutation is applied after the crossover, and thus operator dependencies exist.

References

[1] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 1187–1194.

[2] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms*, ser. Foundations of Genetic Algorithms, L. D. WHITLEY, Ed. Elsevier, 1993, vol. 2, pp. 187–202.

[3] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of North American Fuzzy Information Processing*. IEEE, 1996, pp. 519–523.

[4] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007.

[5] D. Hadka and P. Reed, "Borg: An auto-adaptive many-objective evolutionary computing framework," *Evolutionary Computation*, vol. 21, no. 2, pp. 231–259, 2013.

[6] R. Mallipeddi, S. Mallipeddi, and P. N. Suganthan, "Ensemble strategies with adaptive evolutionary programming," *Inf. Sci.*, vol. 180, no. 9, p. 1571–1581, May 2010.

[7] R. Mallipeddi, P. Suganthan, Q. Pan, and M. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011, the Impact of Soft Computing for the Progress of Artificial Intelligence.

[8] B. Y. Qu, P. Gouthanan, and P. N. Suganthan, "Dynamic grouping crowding differential evolution with ensemble of parameters for multimodal optimization," in *Swarm, Evolutionary, and Memetic Computing*, B. K. Panigrahi, S. Das, P. N. Suganthan, and S. S. Dash, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 19–28.

[9] Q. Pan, P. N. Suganthan, and M. F. Tasgetiren, "A harmony search algorithm with ensemble of parameter sets," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1815–1820.

[10] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Persp.*, vol. 3, pp. 43–58, 2016.

[11] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[12] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. I–115–I–123.

[13] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary Computation*, vol. 10, no. 4, pp. 371–395, 2002.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[15] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[16] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, Aug 2014.

[17] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[18] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[19] S. Mittal, D. K. Saxena, K. Deb, and E. Goodman, Michigan State University, East Lansing, USA, Tech. Rep. COIN Report Number 2020005, 2020.

[20] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization: Theoretical advances and applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. London: Springer London, 2005, pp. 105–145.

[21] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino, "Ga-based decision support system for multicriteria optimization," in *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, vol. 2, 1995, pp. 1556–1561 vol.2.

[22] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural optimization*, vol. 10, no. 2, pp. 94–99, Oct. 1995.