An Elitist Polynomial Mutation Operator for improved performance of MOEAs in Computer Networks

K. Liagkouras, K. Metaxiotis
Decision Support Systems Laboratory, Department of Informatics, University of Piraeus,
80, Karaoli & Dimitriou Str., 18534 Piraeus, Greece

***Abstract*—Polynomial mutation has been utilized in evolutionary optimization algorithms as a variation operator. In previous work on the use of evolutionary algorithms for solving multiobjective problems, two versions of polynomial mutations were introduced. In this study we will examine the latest version of polynomial mutation, the highly disruptive, which has been utilised in the latest version of NSGA-II. This paper proposes an elitist version of the highly disruptive polynomial mutation. The experimental results show that the proposed elitist polynomial mutation outperforms the existing mutation mechanism when applied in a well known evolutionary multiobjective algorithm (NSGA-II) in terms of hypervolume, spread of solutions and epsilon performance indicator.**

***Keywords*—Multi-objective optimization; evolutionary algorithms; mutation.**

I. INTRODUCTION

In multi-objective optimization problems (MOPs) there are many objectives to be optimized at the same time. MOPs can be formally defined as $Min\ (f_1(x), f_2(x), \dots, f_M(x))$ where $x_i^{(L)} \le x_i \le x_i^{(U)}, i = 1, 2, \dots, n.$ A feasible solution to this MOP would be a n-variable solution vector x, which satisfies the variables bounds. Moreover, the solution set of a MOP should satisfy the Pareto Efficiency.

Multiobjective Evolutionary Algorithms (MOEAs) have been used for solving MOPs. The typical MOEA utilize three basic operators: selection, crossover and mutation. The highly disruptive polynomial mutation has been utilized as a major variation operator for MOEAs solving MOPs. It was first introduced by the Deb & Tiwari [4], and implemented in the latest version of NSGA-II [3]. The polynomial mutation is called highly disruptive because of the high amount of disruptiveness that can make to a decision variable when perturbed. The purpose of this work is to introduce a variation of the highly disruptive polynomial mutation (PLM) called Elitist Polynomial Mutation (EPLM) that produces better results. The rest of the paper is structured as follows. In Section II, a description of the highly disruptive polynomial mutation (PLM) is given and in section III the proposed elitist polynomial mutation (EPLM) is presented. The experimental environment is presented in Section IV. In Section V and VI, the performance metrics and the test problems are presented respectively. Finally, in section VII the results are analyzed and conclude the paper.

1. K. Liagkouras is a PhD Candidate at the DSS Lab, Department of Informatics, University of Piraeus, (kliagk@unipi.gr).
2. K. Metaxiotis is an Assistant Professor at the Department of Informatics, University of Piraeus, (kmetax@unipi.gr).

II. POLYNOMIAL MUTATION

Mutation operators have been utilized extensively in MOEAs as solution variation mechanisms. Mutation operators assist to the better exploration of the search space. Different approaches have been proposed depending on the representation used in MOEAs such as binary or real values. For MOEAs solving MOPs, Deb & Goyal [2] proposed a variation mechanism called polynomial mutation (PLM). This operator was later improved in [4].

In polynomial mutation as introduced by [4] each decision variable x_i, can take prices in the interval: $x_i^{(L)} \le x_i \le x_i^{(U)}, i = 1, 2, \dots, n$. Moreover, each decision variable has a probability P_m to be perturbed. For each decision variable, a random value rand is drawn. If rand $\le P_m$ then using the algorithm described in Fig 1, a mutated variable obtains its new value. As we can see from Fig. 1, if random value is r ≤ 0.5 it samples to the left hand side (region between X_{Low} and X_i), otherwise if r > 0.5 it samples to the right hand side (region between X_i and X_{Upper}). The algorithm also calculates the δ_q value to be used in getting the variable its new value. As we can see from the presented algorithm the polynomial mutation operator allows us to sample the entire search space of the decision variable even though the value to be mutated is close to one of the boundaries (X_{Low} - X_{Upper}). Moreover, because PLM allows big jumps in the search space of decision variable, the optimization process has better chances of escaping from local optima and can modify a solution when on the boundary. However, high disruption levels might not be good for achieving smooth approximation of the Pareto front. For instance, if a solution is near an

978-1-4673-5775-3/13/$31.00 ©2013 IEEE

optimal solution then large jumps in the decision space might not be an efficient way of discovering other optimal solutions too. The highly disruptive PLM has been applied in the latest version of NSGA-II [3] and a java implementation of PLM algorithm provided by the jMetal framework [3].

```
rand ⟶ [0, 1];
if (rand <= mutation_probability) then
    δ1 = (Xp − Xl)/(Xu − Xl)        δ2 = (Xu − Xp)/(Xu − Xl)

    r ⟶ [0, 1];
    if (r<= 0.5) then
        dq = 2r + (1 − 2r)(1 − d1)^(ηm+1)]^(1/(ηm+1))
    else
        dq = 1 − [2(1 − r) + 2(r − 0.5)(1 − d2)^(ηm+1)]^(1/(ηm+1))
    end if
```

Fig. 1.Polynomial Mutation (PLM) Pseudo code

III. ELITIST POLYNOMIAL MUTATION (EPLM)

In EPLM variation mechanism when a parent solution is introduced into the system, its decision variables (e.g. in ZDT1 problem each solution consists of 30 decision variables) are tested against the fitness functions.

Each parent solution of the ZDT1 problem consists of 30 decision variables

Parent solution	x_1	x_2	x_3	x_4	x_5	...	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}

Then, the fitness of the various decision variables is normalized. Because, we deal with MOPs, we apply an aggregate objective function with equal weights for each fitness function. In the case of ZDT1 problem the weight of each objective function is obviously 0.50 as we deal with a bi-objective problem.

As soon as we formulate the normalized aggregate fitness evaluation for each decision variable, we arrange these variables into an array, according to their corresponding aggregate objective fitness (from best performing decision variable to worst). Thus, in position zero of the array resides the best performing decision variable and in the last position of the array resides the worst performing variable.

Highest Fitness variable ↓ *Lowest Fitness variable* ↓

Sorted variables	x_{23}	x_5	x_{28}	x_7	x_{10}	.....	x_{30}	x_8	x_{14}	x_2	x_9	x_{22}

Then we apply the polynomial mutation (PLM) mechanism introduced by Deb and Tiwari [4] but with a crucial variation. In the original implementation of PLM, the mutation mechanism is triggered if rand $\leq P_{m,}$, where rand is a random number in the range [0, 1] and $P_{m,}$ is the mutation probability, which is equal to 1/n (n is the number of real variables). Thus, for example if a random number rand smaller or equal to the mutation probability (rand $\leq P_m$) is generated in the fifth attempt, then the fifth decision variable will be mutated. However, in EPLM implementation, instead of picking up the fifth ordinal decision variable (i.e. x_5), we choose the fifth higher fitness variable to be mutated. At this point we should remind that the various decision variables are sorted according to their corresponding fitness.

Selected variable for mutation ↓

sorted variables	x_{23}	x_5	x_{28}	x_7	x_{10}	.....	x_{30}	x_8	x_{14}	x_2	x_9	x_{22}

As soon as the variable has been selected, then it is variated according to the polynomial mutation mechanism [4], as follows:

$$\delta_1 = \frac{X_p - X_l}{X_u - X_l} \qquad \delta_2 = \frac{X_u - X_p}{X_u - X_l}$$

$$\delta_q = \begin{cases} [2r + (1-2r)(1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1, & if\ r \leq 0.5, \\ 1 - [2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}, & if\ r > 0.5 \end{cases}$$

Thereafter, the mutated variable value X_C is computed as follows: $X_c = X_p + \delta_q(X_u - X_l)$

X_p is the parent solution (a real variable)
X_c is the child solution (a real variable)
X_l is the lower bound of the solution variable
X_u is the upper bound of the solution variable
r is a random number uniformly distributed in [0, 1] , and
η_m is the index for polynomial mutation

Finally, in EPLM instead of replacing the initial variable with the mutated one, we follow a different methodology. Specifically, we locate which variable resides in the *n-count* (i.e. 29 – 4 = 25th) position of the sorted variables.

x_8 variable resides in the n – count position ↓

sorted variables	x_{23}	x_5	x_{28}	x_7	x_{10}	.....	x_{30}	x_8	x_{14}	x_2	x_9	x_{22}

In this case the x_8 variable resides in the *n – count* position of the sorted decision variables.

Next, the mutated variable: $M(x_{10})$ is reinserted in the parent solution at the position of x_8 variable. Thus, is formulated the child solution.

The $M(x_{10})$ is reinserted in the position of x_8 variable ↓

Child solution	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$M(x_{10})$	..	x_{28}	x_{29}	x_{30}

The Elitist Polynomial Mutation mechanism that just presented allows the better exploration, of solution space, around the already known good solutions. Through the elitism mechanism introduced in EPLM, the MOEAs can escape local optima solutions and at the same time the exploratory effort is focused near already known good solutions. Below, we provide empirical evidence comparing the performance of a well known evolutionary multiobjective algorithm (NSGA-II) under the classical polynomial mutation (PLM) and under the elitist polynomial mutation (EPLM) mechanism proposed in this paper.

```
Begin
mutation_probability = 1/N;
ηm = distribution index;
 for n=0 to N; (length of variables)
     Evaluate Fitness (variable)
 end
     Sort variables (Fitness)
          count = 0;
          for n=0 to N; (length of variables)
     rand ⟶ [0, 1];
       if (rand <= mutation_probability) then
          index_of_variable_A = Sorted_ variables [count];
            xp  = getValue(index_of_variable_A);
            xl  = getLowerBound(index_of_variable_A);
            xu  = getUpperBound(index_of_variable_A);
              δ1 = (Xp − Xl)/(Xu − Xl)      δ2 = (Xu − Xp)/(Xu − Xl)
```

```
r  ⟶ [0, 1];
if (r<= 0.5) then
    d_q = 2r + (1 − 2r)(1 − d_1)^(η_m+1)]^(1/(η_m+1))
else
    d_q = 1 − [2(1 − r) + 2(r − 0.5)(1 − d_2)^(η_m+1)]^(1/(η_m+1))
end if
   X_c = X_p + d_q( X_u − X_l)
   if ( X_c < X_l ) then
        X_c = X_l ;
    end if
    if ( X_c > X_u ) then
        X_c = X_u ;
    end if
   variable_B = Sorted_ variables [N - count];
   index_of_variable_B = Parent_Solution [find: variable_B];
    Child_Solution =  Parent_Solution.setValue(index_of_variable_B ,  X_c);
     end if
  count++;
end
```

Fig. 2. Elitist Polynomial Mutation (EPLM) Pseudo code

IV. EXPERIMENTAL ENVIRONMENT

A. Parameter Setup

The jMetal [5] framework has been used to test the EPLM against the PLM variation mechanism. JMetal is a Java based framework for multiobjective optimisation and has been used in several studies of multiobjective metaheuristics [7]. We carry out the test by applying the proposed EPLM mechanism to Nondominated Sorting Genetic Algorithm II (NSGA-II) [3] and comparing the results with the results derived by the same MOEA using its typical configuration (i.e. with the PLM operator).

In all tests we use binary tournament and simulated binary crossover (SBX) [1] as selection and crossover operators, respectively. The crossover probability is $P_c = 1.0$ and mutation probability is $P_m = 1/n$, where n is the number of decision variables. The distribution indices for the crossover and mutation operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. Population size is set to 100, using 25,000 function evaluations with 100 independent runs.

V. PERFORMANCE METRICS

A. Hypervolume

Hypervolume [6], also known as Lebesgue measure or S metric, is an indicator of both the convergence and diversity of an approximation set. Thus, given a set S containing m points in n objectives, the hypervolume of S is the size of the portion of objective space that is dominated by at least one point in S. The hypervolume of S is calculated relative to a reference point which is worse than (or equal to) every point in S in every objective. The greater the hypervolume of a solution the better considered the solution. One of the main advantages of hypervolume [9] is that it is able to capture in a single number both the closeness of the solutions to the optimal set and, to some extent, the spread of the solutions across objective space.

B. Spread

Deb et al. [3] introduced the spread of solutions (Δ) as another indicator of the quality of the derived set of solutions. As its name reveals this indicator examines whether or not the solutions span the entire Pareto optimal region. Moreover, it calculates the Euclidean distance between the consecutive solutions in the obtained non-dominated set of solutions. Then it calculates the average of these distances. After that, from the obtained set of non-dominated solutions the extreme solutions are calculated. Finally, using the following metric it calculates the nonuniformity in the distribution.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}}$$

Where d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the obtained nondominated set. The parameter $\bar{d}$ is the average of all distances d_i, i = 1, 2, ... , (N - 1), where N is the number of solutions on the best nondominated front.

C. Epsilon Indicator I_ε

Zitzler et al. [11] introduced the epsilon indicator (I_ε). There are two versions of epsilon indicator the multiplicative and the additive. In this study we use the unary additive epsilon indicator as it has been implemented in jMetal framework. To state briefly the basic usefulness of epsilon indicator of an approximation set A ($I_{\varepsilon+}$) is that it provides the minimum factor ε by which each point in the real front R can be added such that the resulting transformed approximation set is dominated by A.

VI. THE TEST PBOBLEMS

The Zitzler-Deb-Theile (ZDT) test suite [10] is widely used for evaluating algorithms solving MOPs. The following three bi-objective MOPs named ZDT1, ZDT2, ZDT3 were used for comparing the proposed EPLM operator against the polynomial mutation (PLM). They have been used extensively for testing MOEAs and their Pareto front shapes are convex, nonconvex and disconnected. ZDT1, ZDT2 and ZDT3 use 30 decision variables.

A. Zitzler-Deb-Thiele's function N.1 problem:

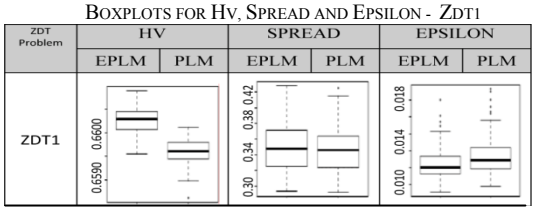
$$Min = \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \end{cases} \quad for\ 0 \le x_i \le 1\ and\ 1 \le i \le 30$$

TABLE I
MEAN, STD, MEDIAN AND IQR FOR HV, SPREAD AND EPSILON - ZDT1

Problem		NSGAII EPLM	NSGAII
ZDT1	HV. Mean and Std	6.60e-01$_{2.5e-04}$	6.60e-01$_{2.8e-04}$
	HV. Median and IQR	6.60e-01$_{2.8e-04}$	6.60e-01$_{3.9e-04}$
	SPREAD. Mean and Std	3.51e-01$_{2.9e-02}$	3.45e-01$_{2.8e-02}$
	SPREAD. Median and IQR	3.53e-01$_{4.4e-02}$	3.46e-01$_{4.0e-02}$
	EPSILON. Mean and Std	1.26e-02$_{1.9e-03}$	1.29e-02$_{2.1e-03}$
	EPSILON. Median and IQR	1.23e-02$_{2.4e-03}$	1.28e-02$_{2.8e-03}$

TABLE II

BOXPLOTS FOR HV, SPREAD AND EPSILON - ZDT1



The results in the Tables I - IX have been produced by using JMetal [5] framework. Table I presents the results of ZDT1 test function. Specifically, it presents the mean, standard deviation (STD), median and interquartile range (IQR) of all the independent runs carried out for Hypervolume (HV), Spread (Δ) and Epsilon indicator respectively.

Clearly, regarding the HV [9] indicator the higher the value (i.e. the greater the hypervolume) the better the computed front. HV considered by many researchers in the field [6], [11] the most important indicator as it captures in a single number both the closeness of the solutions to the optimal set and to a certain degree, the spread of the solutions across objective space. The second indicator the Spread (Δ) [3] examines the spread of solutions across the pareto front. The smaller the value of this indicator, the better the distribution of the solutions. This indicator takes a zero value for an ideal distribution of the solutions in the Pareto front. The third indicator, the Epsilon [8] is a measure of the smaller distance that a solution set A, needs to be changed in such a way that it dominates the optimal Pareto front of this problem. Obviously the smaller the value of this indicator, the better the derived solution set.

Tables II, V and VIII use boxplots to present graphically, the performance of NSGA-II under two different configurations, EPLM and PLM respectively, for the three performance indicators, namely: HV, Spread and Epsilon. Boxplot is a convenient way of depicting graphically groups of numerical data. Finally, Tables III, VI and IX present if the results of NSGA-II derived under the two different configurations (EPLM and PLM) are statistically significant or not. For that reason, we use the Wilcoxon rank-sum test as it is implemented by the jMetal framework [5]. In Tables III, VI and IX three different symbols are used. In particular "–" indicates that there is not statistical significance between the algorithms. "▲" means that the algorithm in the row has yielded better results than the algorithm in the column with confidence and "▽" is used when the algorithm in the column is statistically better than the algorithm in the row.

TABLE III

WILCOXON TEST IN ZDT1 FOR HV, SPREAD AND EPSILON

Problem			NSGAII
ZDT1	NSGAII'EPLM	HV. Mean and Std	▲
		HV. Median and IQR	▲
		SPREAD. Mean and Std	–
		SPREAD. Median and IQR	–
		EPSILON. Mean and Std	–
		EPSILON. Median and IQR	–

B. Zitzler-Deb-Thiele's function N.2 problem:

$$Min = \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i \\ h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2 \end{cases} \quad for\ 0 \le x_i \le 1\ and\ 1 \le i \le 30$$

TABLE IV

MEAN, STD, MEDIAN AND IQR FOR HV, SPREAD AND EPSILON - ZDT2

Problem		NSGAII'EPLM	NSGAII
ZDT2	HV. Mean and Std	3.27e-01$_{2.0e-04}$	3.26e-01$_{2.6e-04}$
	HV. Median and IQR	3.27e-01$_{2.9e-04}$	3.26e-01$_{3.4e-04}$
	SPREAD. Mean and Std	3.58e-01$_{2.6e-02}$	3.52e-01$_{3.2e-02}$
	SPREAD. Median and IQR	3.56e-01$_{3.1e-02}$	3.56e-01$_{4.7e-02}$
	EPSILON. Mean and Std	1.24e-02$_{2.0e-03}$	1.31e-02$_{2.3e-03}$
	EPSILON. Median and IQR	1.21e-02$_{2.7e-03}$	1.23e-02$_{2.7e-03}$

TABLE V

BOXPLOTS FOR HV, SPREAD AND EPSILON - ZDT2

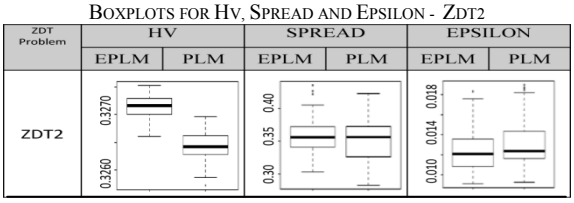


TABLE VI

WILCOXON TEST IN ZDT2 FOR HV, SPREAD AND EPSILON

Problem			NSGAII
ZDT2	NSGAII'EPLM	HV. Mean and Std	▲
		HV. Median and IQR	▲
		SPREAD. Mean and Std	–
		SPREAD. Median and IQR	–
		EPSILON. Mean and Std	▲
		EPSILON. Median and IQR	▲

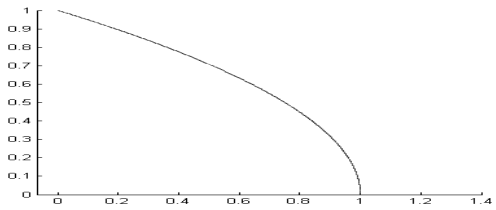


Fig. 3. ZDT2 Pareto front, formulated by using the NSGA-II combined with EPLM as variation operator

C. Zitzler-Deb-Thiele's function N.3 problem

$$Min = \begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h(f_1(x), g(x)) \\ g(x) = 1 + \frac{9}{29}\sum_{i=2}^{30} x_i \\ h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right)\sin(10\pi f_1(x)) \end{cases} \quad for\ 0 \le x_i \le 1\ and\ 1 \le i \le 30$$

TABLE VII
MEAN, STD, MEDIAN AND IQR FOR HV, SPREAD AND EPSILON - ZDT3

Problem		NSGAII'EPLM	NSGAII
ZDT3	HV. Mean and Std	$5.14e\text{-}01_{7.2e\text{-}03}$	$5.15e\text{-}01_{1.4e\text{-}04}$
	HV. Median and IQR	$5.15e\text{-}01_{1.6e\text{-}04}$	$5.15e\text{-}01_{1.9e\text{-}04}$
	SPREAD. Mean and Std	$7.48e\text{-}01_{1.7e\text{-}02}$	$7.44e\text{-}01_{1.5e\text{-}02}$
	SPREAD. Median and IQR	$7.45e\text{-}01_{2.3e\text{-}02}$	$7.42e\text{-}01_{1.7e\text{-}02}$
	EPSILON. Mean and Std	$1.34e\text{-}02_{2.9e\text{-}02}$	$8.04e\text{-}03_{1.5e\text{-}03}$
	EPSILON. Median and IQR	$7.94e\text{-}03_{2.1e\text{-}03}$	$7.86e\text{-}03_{1.5e\text{-}03}$

TABLE VIII
BOXPLOTS FOR HV, SPREAD AND EPSILON - ZDT3

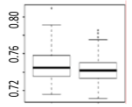
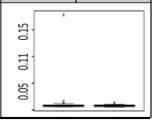
ZDT Problem	HV		SPREAD		EPSILON	
	EPLM	PLM	EPLM	PLM	EPLM	PLM
ZDT3						

TABLE IX
WILCOXON TEST IN ZDT3 FOR HV, SPREAD AND EPSILON

Problem			NSGAII
ZDT3	NSGAII'EPLM	HV. Mean and Std	▲
		HV. Median and IQR	▲
		SPREAD. Mean and Std	—
		SPREAD. Median and IQR	—
		EPSILON. Mean and Std	—
		EPSILON. Median and IQR	—

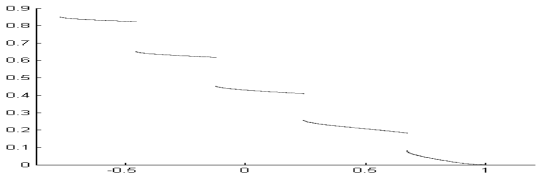


Fig. 4. ZDT3 Pareto front, formulated by using the NSGA-II combined with EPLM as variation operator

VII. ANALYSIS OF THE RESULTS

In this section, we analyze the results obtained by applying the Elitist Polynomial Mutation (EPLM) operator and the Polynomial Mutation (PLM) operator respectively to the NSGA-II. Three well known performance indicators i.e. HV, Spread and Epsilon of MOEAs have been applied to access the quality of the proposed variation operator.

Examining the results (Table I, IV and VII) of the first indicator, the Hypervolume, which is considered by many in the relevant literature [6], [11] as the most representative indicator of the performance of a MOEA we notice that the NSGA-II performs better with the application of EPLM operator in all three test functions (ZDT1, ZDT2, ZDT3) examined, compared with results derived by applying the classical PLM operator. The boxplot (Table II, V and VIII) provides a graphical representation of the relevant results and the Wilcoxon rank-sum test (Table III, VI and IX) confirms that the NSGA-II with the EPLM operator yields better results with confidence, than the classical configuration of NSGA-II with the PLM operator. Regarding the Spread indicator, the Wilcoxon rank-sum test indicates that there is not statistical significance between the two variation operators. Meaning that neither variation operator outperforms the other. Finally, by observing the relevant results about the Epsilon indicator, we see that in the case of ZDT2 the proposed EPLM operator outperform the results derived by the classical PLM operators. In the remaining cases (ZDT1 and ZDT3) the two variation operators generate not statistically significant results.

To conclude, the proposed methodology (EPLM) generates better results in all test functions examined regarding the HV indicator. It also produce better results in one test function (ZDT2) regarding the Epsilon indicator and not worse results than the classical PLM operator for the rest of the cases.

REFERENCES

[1] Deb, K. & Agrawal, R.B. (1995) Simulated binary crossover for continuous search space, Complex Systems 9 (2) 115–148.

[2] Deb, K. & Goyal, M. (1996). A combined genetic adaptive search (geneas) for engineering design, Computer Science and Informatics 26 (4) 30–45.

[3] Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA, II. IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182–197.

[4] Deb, K. & Tiwari, S. (2008) Omni-Optimizer: A Generic Evolutionary Algorithm for Single and Multi-Objective Optimization. European Journal of Operational Research, Vol. 185.

[5] Durillo, J. J. & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization, Advances in Engineering Software 42, 760–771

[6] Emmerich, M., Beume, N. & Naujoks, B. (2005). An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 62–76. Springer.

[7] Herbawi, W. & Weber, M. (2011). Evolutionary Multiobjective Route Planning in Dynamic Multi-hop Ridesharing.Evolutionary Computation in Combinatorial Optimization Lecture Notes in Computer Science, Volume 6622/2011, 84-95

[8] Knowles, J., Thiele, L. & Zitzler. E. (2006). A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

[9] Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

[10] Zitzler, E., Deb, K. & Thiele., L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

[11] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M. & Da Fonseca, V.G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.