International Journal on Artificial Intelligence Tools
Vol. 20, No. 1 (2011) 209–219

DOI: 10.1142/S0218213011000097

A DYNAMIC POLYNOMIAL MUTATION FOR EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

MOHAMMAD HAMDAN
Computer Science, Faculty of IT, Yarmouk University, Irbid 21163, Jordan
hamdan@yu.edu.jo

Received 1 March 2009
Accepted 17 July 2010

Polynomial mutation is widely used in evolutionary optimization algorithms as a variation operator. In previous work on the use of evolutionary algorithms for solving multi-objective problems, two versions of polynomial mutations were introduced. The first is non-highly disruptive that is not prone to local optima and the second is highly disruptive polynomial mutation. This paper looks at the two variants and proposes a dynamic version of polynomial mutation. The experimental results show that the proposed adaptive algorithm is doing well for three evolutionary multiobjective algorithms on well known multiobjective optimization problems in terms of convergence speed, generational distance and hypervolume performance metrics.

Keywords: Multi-objective optimization; evolutionary algorithms; mutation.

1. Introduction

In multi-objective optimization problems (MOPs) there are many (conflicting) objectives to be optimised at the same time. MOPs can be formally defined as:

$$Min[f_1(\vec{x}), f_2(\vec{x}), \ldots, f_d(\vec{x})] \tag{1}$$

where $\vec{x} \in \Omega \subseteq \Re^n$ is a vector of decision variables and f_i are the objective functions. The solution for a MOP is a set called *non-dominated solutions*. Here no solution dominates others in this set. The Pareto dominance relation denoted by "$\prec$" can be defined as follows[1]:

$$\begin{aligned}(\vec{x_1} \prec \vec{x_2}) \Leftrightarrow \forall i \in \{1, \ldots, d\}, f_i(\vec{x_1}) \leq f_i(\vec{x_2}) \\ and \quad \exists i \ \in \{1, \ldots, d\}, f_i(\vec{x_1}) < f_i(\vec{x_2}).\end{aligned} \tag{2}$$

The solution set satisfying the Pareto dominance relation is called *Pareto-optimal set*. When plotted in the objective space they define the true Pareto-Front (PF_{true}). An optimization algorithm needs to find PF_{known} that approximates PF_{true}.

Genetic algorithms (GAs)[10] have been used for solving MOPs. They use three major operators: parent selection, crossover and mutation. The polynomial mutation[4] has been used as the major mutation operator for GAs solving MOPs. It was first introduced and implemented in NSGA[a] and NSGAII.[5]

In the published literature, there are two different implementations for polynomial mutation under a single name. The difference between both versions can be noticed be observing the corresponding source code. The two versions differ in the amount of disruptiveness they can make to a decision variable when perturbed. The purpose of this work is to illustrate the difference between the two algorithms and introduce a new polynomial mutation that hybridises both.

2. The Polynomial Mutation

The original polynomial mutation (non-highly disruptive)[4,2] was first used in NSGA,[13] early versions of NSGA-II[5] and DEME[b] project.[9] As illustrated in Figure 1, if the decision variable X is to be perturbed then its new value could be somewhere either in region a (sample to left hand side) or region b (sample to right hand side). The mutation works as follows (described in Algorithm 1). For each decision variable x_i, box constraints are defined in $[x_i^{Lower}, x_i^{Upper}]$. Each decision variable has a probability P_m to be perturbed. For each decision variable, a random value r is drawn. If $r \leq P_m$ then the procedure first finds the difference between the variable and the lower boundary called δ_1 and the variable and upper boundary called δ_2. Both differences are divided by $(X_{Upper} - X_{Lower})$. The smallest difference is called δ. Another random value r is drawn. If $r \leq 0.5$ then sample to the left hand side; otherwise to the right hand side.

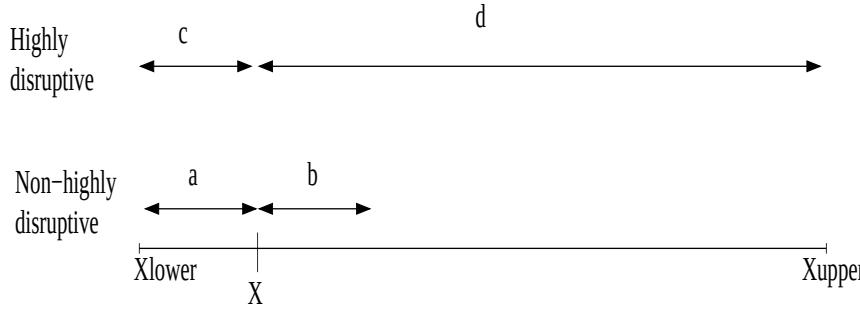

Fig. 1. The sampling space for the non-highly and highly disruptive polynomial mutations.

Notice that if the value of the decision variable to be mutated is close to one of its boundaries (i.e. the δ value is very small) then the mutation becomes useless

[a]NSGA stands for Non-dominated Sorting Genetic Algorithm.
[b]DEME stands for DistributEd MEtaheuristics. It is a project for the design and development of parallel and distributed multi-objective metaheuristic algorithms.

p $\leftarrow$ value in the range [0.1,0.9] specified by Algorithm 2
u $\leftarrow$ U[0,1]
if $u > p$ **then**
  mut $\leftarrow$ 1
end
else
  mut $\leftarrow$ 2
end
i $\leftarrow$ 0
repeat
  r $\leftarrow$ U[0,1]
  if $r \leq P_m$ **then**
    $\delta_1 \leftarrow \frac{X_i - X_i^{Lower}}{X_i^{Upper} - X_i^{Lower}}$, $\delta_2 \leftarrow \frac{X_i^{Upper} - X_i}{X_i^{Upper} - X_i^{Lower}}$
    r $\leftarrow$ U[0,1]
    if $u > p$ **then**
      $\delta \leftarrow min\{\delta_1, \delta_2\}$
    end
    else
      $\delta \leftarrow \begin{cases} \delta_1 \text{ if } r \leq 0.5 \\ \delta_2 \text{ otherwise} \end{cases}$
    end
    $$\delta_q \leftarrow \begin{cases} [(2r) + (1-2r)* \\ (1-\delta)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1 & \text{if } r \leq 0.5 \\ 1 - [2(1-r) + 2.(r-0.5)* \\ (1-\delta)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} & \text{otherwise} \end{cases}$$
    $X_i \leftarrow X_i + \delta_q.(X_i^{Upper} - X_i^{Lower})$
  end
until $i_{++} == n$

Algorithm 1. Hybrid polynomial mutation.

and the algorithm becomes not prone to local optima. Nonetheless, small values are needed sometimes to get better approximations of PF_{true}.

The modified polynomial mutation (highly disruptive)[7] has been used in the jMetal Toolkit[8] and the latest version of NSGA-II.[5] It differs from its predecessor in the choice of δ. As it can be seen in Figure 1, using the highly disruptive mutation, the decision variable X can have its new value somewhere either in region c (sample to left hand side) or region d (sample to right hand side). Here, if we are sampling to the left hand side then the value of δ is δ_1 otherwise it is δ_2. This gives the possibility of doing larger jumps in the search space and thus escaping from local optima.

In the new hybrid polynomial mutation, both versions of mutation can used according to a probability. Here we introduced probability variable p. Different disruption levels can be quantified using different p values When $p = 0.0$ then only

```
Offspring ← PolyNomialMutation(Parent)
if Offspring ≺ Parent then
    if mut == 1 then
        Sucess1++
    end
    else
        Sucess2++
    end
end
if Evaluations % M == 0 then
    if Success1 > Success2 then
        p ← p - 0.1, if (p < 0.1) p ← 0.1
    end
    else
        p ← p + 0.1,  if (p > 0.9) p ← 0.9
    end
    Sucess1 ← 0,  Sucess2 ← 0
end
```

Algorithm 2. Proposed dynamic polynomial mutation.

the non-highly disruptive Polynomial mutation is used. Also setting $p = 1.0$ then only the highly disruptive is used. In other words, with probability $1 - p$ use the non-highly disruptive an with probability p use the highly disruptive.

To automate the process of setting the p value a new technique can be used within an algorithm that uses polynomial mutation. The Dynamic Polynomial Mutation (DPM) is described in Algorithm 2. The variable *mut* is used to differentiate between the first and the second versions of the Polynomial mutation. $Success_1$ and $Success_2$ are counters for counting the number of successful mutations using the first and second types of mutations, respectively. The current number of function evaluations performed so far is stored in *Evaluations*.

DPM is described next. After an offspring is generated from parent solution by mutation check if the offspring dominates the parent. If offspring $\prec$ parent then consider this as a successful mutation and increment the corresponding counter. Here, each mutation has its counter. After M function evaluations find which counter is bigger. In case the non-highly disruptive mutation gave more solutions that dominated the parents then move the p value toward the successful mutation. Thus giving more probability in using the non-highly disruptive mutation. Otherwise move the p value towards the highly disruptive mutation.

The idea is that the mutation that gives more solutions that dominate the parents is giving better improvement for the EMOA. This could be in escaping from local optima or generating better solutions. Initially we set $p \leftarrow 0.5$. This procedure has to be inserted in the EMOA after it uses the Polynomial mutation and evaluating the new offspring. The value of M depends on the complexity of the problem (i.e. number of decision variables). In the experiments it was set to 30 for

problems with small number of variables and 100 for problems with large number of variables. It is important to give enough number of function evaluations for the EMOA to generate new solutions that dominate the parents.

3. Experimental Environment

The jMetal[8] toolkit has been used to test the proposed dynamic polynomial mutation. It is a java based framework for multiobjective optimizations. Other researchers have also used the toolkit for studying multiobjective metaheuristics.[11] The jMetal toolkit provides implementations for the Nondominated Sorting Genetic Algorithm II (NSGA-II),[5] the Strength Pareto Evolutionary Algorithm (SPEA2)[16] and the MultiObjective Cellular Genetic Algorithm (MOCell).[12] These algorithms are well known for solving MOPs have been used for the experimental study. All of the genetic algorithms use binary tournament, simulated binary crossover (SBX)[3] and polynomial mutation as selection, crossover and mutation operators, respectively.

The Zitzler-Deb-Theile (ZDT) test suite.[15] The following five bi-objective MOPs named ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 were used for evaluating the disruptions level of Polynomial mutation and the new dynamic polynomial mutation. They are well understood and their Pareto front shapes are convex, non-convex, disconnected, multi-modal and non-uniformly spaced respectively. ZDT1, ZDT2 and ZDT3 use 30 decision variables while ZDT4 and ZDT6 use 10 decision variables. Also the scalable Deb-Thiele-Laumanns-Zitzler (DTLZ) MOPs[6] were used. We set 100 decision variables and use the bi-objective formulation of DTLZ1, DTLZ2, DTLZ3 and DTLZ6.

The crossover probability is $P_c = 0.9$ and mutation probability is $P_m = 1/n$, where n is number of decision variables. The distribution index for the crossover and mutation operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. Population size is set to 100, using 25000 function evaluations (for the DTLZ MOPs we use 1×10^5 function evaluations) with 100 independent runs for all problems.

Two well known metrics were used for measuring the performance of the proposed DPM: the Generational Distance (GD)[14] and the convergence speed (number of function evaluations needed to generate an acceptable Pareto front, i.e. stop when $HV(PF_{known}) \geq 98\% * HV(PF_{true})$). In the experiments we use DPM and different p values.

4. Results

The results reported in this Section address two objectives: (1) the quality of the final obtained solution using the GD metric for different p values and DPM, and (2) the *convergence speed* metric for obtaining an acceptable approximation of PF_{true} using different p values and DPM. Both metrics are to be minimized. The box plots for the GD and *convergence speed* metrics for ZDT MOPs are shown in Figures 2 and 3, respectively.

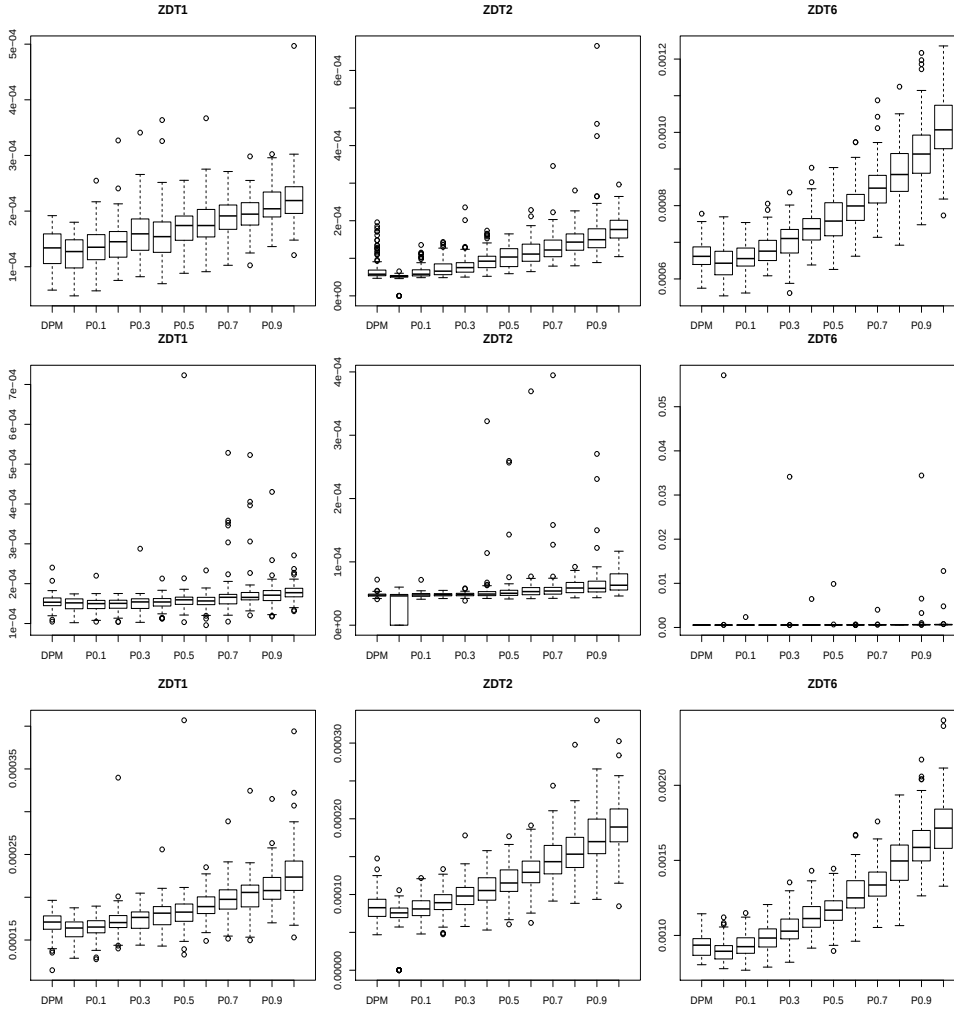

Fig. 2. Box plots for the generational distance metric for ZDT1, ZDT2 and ZDT6 using NSGAII (first row), MOCell (second row) and SPEA2 (third row).

- ZDT1: All EMOAs converge faster (minimum number of function evaluations) when the non-highly disruptive polynomial mutation (i.e. $p = 0.0$) is used. The *convergence speed* progressively decrease (linearly) as p increases towards 1.0. Regarding the GD metric it is clear that the best result is achieved when $p = 0.0$. Regarding DPM, it gave very competitive results to $p = 0.0$ for all EMOAs.
- ZDT2: For all EMOAs, the GD and *convergence speed* metrics progressively decrease as p increases towards 1.0. This concludes that all EMOAs prefer the non-highly disruptive mutation. However, by having a closer look at the box plots for the GD metric (Figure 2) we can find few outlier values especially for

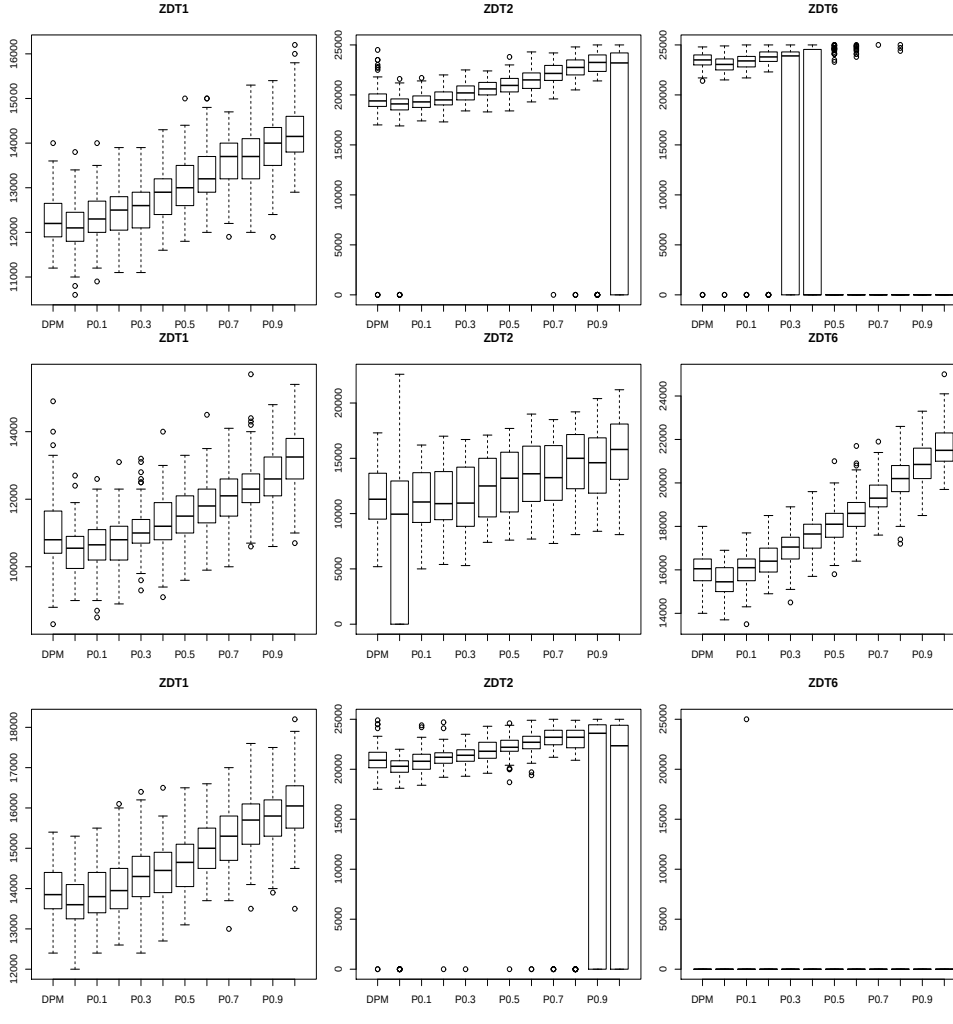

Fig. 3. Box plots for the convergence speed metric for ZDT1, ZDT2 and ZDT6 using NSGAII (first row), MOCell (second row) and SPEA2 (third row).

MOCell. This suggests that the non-highly disruptive polynomial (with $p = 0.0$) is neither resilient to local optima nor prone to premature convergence.

The DPM gave very competitive results for all EMOAs when compared with $p = 0.0$ and outperformed all other p values (i.e. $p \geq 0.1$). In fact, DPM can generate solutions with similar behavior to the non-highly disruptive mutation without the risk of falling in local optima.

- ZDT3 and ZDT4: For ZDT3, the differences in performance for both metrics for all EMOAs are not significant. Nonetheless, there is a slight preference for the non-highly disruptive mutation but it is not that noticeable. Regarding ZDT4,

for all EMOAs when using $p = 0.0$ there is a risk in falling in local optima. The box plots are similar to those of ZDT2. The DPM mutation preferred $p = 0.0$ for ZDT3 and no preference for ZDT4. The results are not shown due to space limitations.

- ZDT6: Here, it is interesting to see that all metrics decrease as p also increases. For all EMOAs, the preference is to use the non-highly disruptive mutation. However, NSGAII fails in getting an acceptable Pareto front when $p \geq 0.4$. SPEA2 failed for all p values in getting an acceptable Pareto front. Also, it is interesting to see that we can save more than 6000 functions evaluation when using DPM mutation instead of the highly disruptive mutation (when $p = 1.0$). Again, DPM gave very competitive results when compared with $p = 0.0$.

In summary, the non-highly disruptive mutation outperforms the highly disruptive Polynomial mutation but has a high risk of failed runs as seen in ZDT2 and ZDT4. However, DPM gave very competitive results to $p = 0.0$. Also it outperforms all values of $p \geq 0.1$ for all MOPS using the three EMOAs without the risk of falling in local optima.

In the next set of experiments, the goal is to see the behavior of the hybrid polynomial mutation and DPM with additional MOPs with large number of variables. The box plots for the GD metric are shown in Figure 4. The *convergence speed* results are not shown as even after 1×10^5 function evaluations none of the EMOAs managed to get $HV(PF_{known}) \geq 98\% * HV(PF_{true})$.

- DTLZ1 and DTLZ3: Interesting results are seen here as the GD metric progressively improves as p moves toward 1.0. This behavior is different from the previous ZDT MOPs where they preferred the non-highly disruptive mutation. This finding could be due to nature of the large search space of DTLZ1 and DTLZ3 where large jumps are preferable. The DPM results for the GD metric are very similar to $p = 1.0$. However, all of the EMOAs failed in finding an acceptable PF_{known} even after using a large number of function evaluations.
- DTLZ2 and DTLZ6: The results for DTLZ2 are not shown since there is no difference as p moves from 0.0 to 1.0. For DTLZ6, the GD metric increases we p moves toward 0.0. In other words, all EMOAs prefer the non-highly disruptive polynomial mutation. Here, all values of p and DPM gave similar behavior in terms of GD and HV metrics. Again DPM results are competitive to those when $p = 0.0$ or 0.1 and outperforms all $p \geq 0.2$ for nearly all EMOAs.

It can be concluded that different MOPs (such as ZDT1, ZDT2, ZDT6 and DTLZ6) prefer the non-highly disruptive mutation with the possibility of finding MOPs (such as ZDT3, ZDT4 and DTLZ2) that prefer neither version or prefer the highly disruptive (such as DTLZ1 and DTLZ3). Using DPM, it can be asserted that the p value is shifted towards the most successful type of mutation to be used for a given MOP.

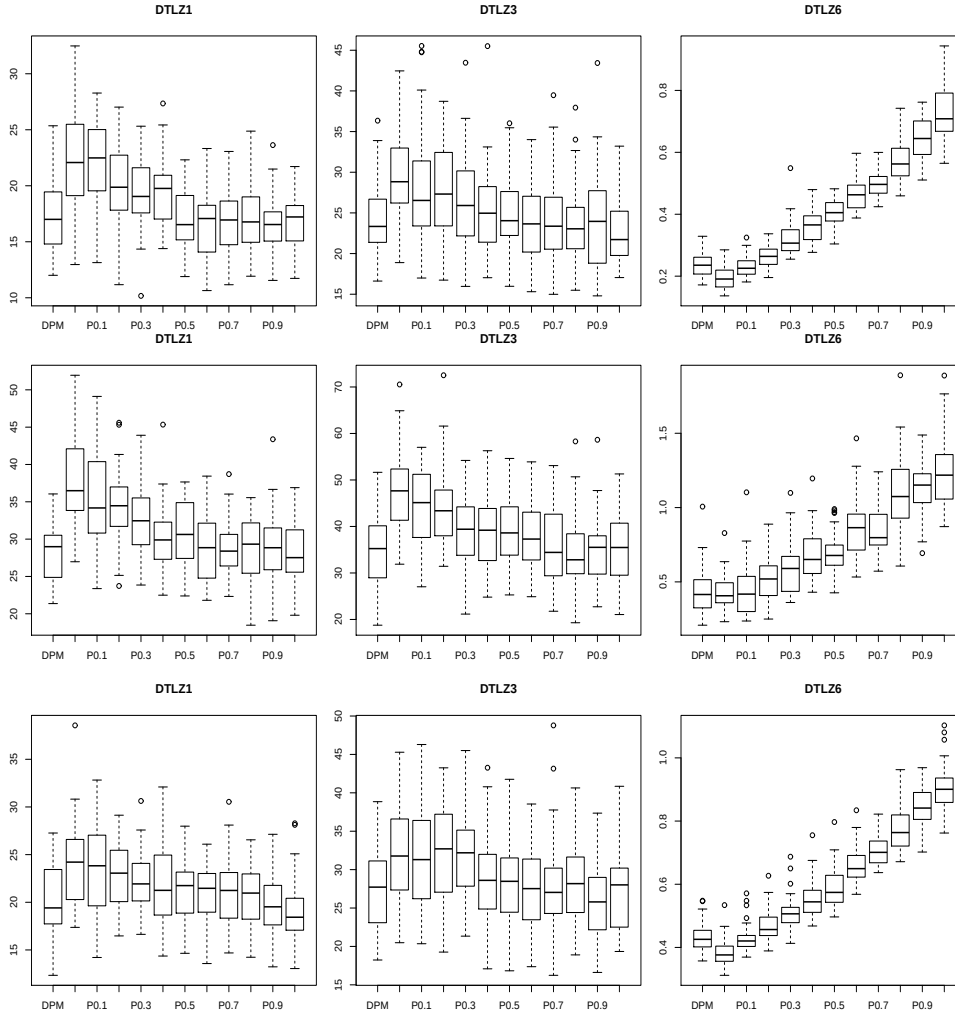

Fig. 4. Box plots for the generational distance metric for DTLZ1, DTLZ3 and DTLZ6 using NSGAII (first row), MOCell (second row) and SPEA2 (third row).

5. Conclusions

The work in this paper has presented a new adaptive mutation called DPM that hybridises both the non-highly disruptive and the highly disruptive polynomial mutation. This version retains the advantages of both algorithms. polynomial mutation. The work was tested on three EMOAs and two well known benchmark problems. The results showed that DPM is an adaptive mutation that can be used with different EMOAs for different MOPs. The results also show that choosing the right mutation is very important for faster convergence and better GD results.

Using DPM, we give chances of using the most suitable type of mutation to be used for a given MOP.

Acknowledgment

The author would like to thank DFG for supporting the research visit. Also thanks to Prof. David Corne/Heriot-Watt University for useful discussion during the development of this work.

References

1. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems* (Kluwer Academic Publishers, New York, May 2002), ISBN 0-3064-6762-3.
2. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms* (Wiley, Chichester, UK, 2001).
3. K. Deb and R. Agrawal, Simulated binary crossover for continous search space, *Complex Systems* **9** (1995) 115–148.
4. K. Deb and M. Goyal, A combined genetic adaptive search (geneas) for engineering design, *Computer Science and Informatics* **26**(4) (1996) 30–45.
5. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA–II, *IEEE Transactions on Evolutionary Computation* **6**(2) (April 2002) 182–197.
6. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, Scalable test problems for evolutionary multiobjective optimization. In L. Jain A. Abraham and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pp. 105–145, Hiedelberg, 2005. Springer.
7. K. Deb and S. Tiwari, Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization, *European Journal of Operational Research* (2008) 185.
8. Juan J. Durillo, Antonio J. Nebro, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba, jMetal: A Java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, December 2006.
9. A. Nebro *et al.*, The deme project., [URL] http://neo.lcc.uma.es/Software/deme/html/. Accessed December 2008.
10. D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Publishing Company, Inc., 1989).
11. Carlos A. Coello Coello Francisco Luna Juan J. Durillo, Antonio J. Nebro, and Enrique Alba, A comparative study of the effect of parameter scalability in multi-objective metaheuristics. In *IEEE Congress on Evolutionary Computing*, Hong Kong, June 2008.
12. Antonio J. Nebro, Juan J. Durillo, Francisco Luna, Bernabé. Dorronsoro, and Enrique Alba, A cellular genetic algorithm for multiobjective optimization. In David A. Pelta and Natalio Krasnogor, editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25–36, Granada, Spain, 2006.
13. N. Srinivas and K. Deb, Multi-objective function optimization using non-dominated sorting genetic algorithms *Evolutionary Computation* **2** (1995) 221–248.

14. David A. Van Veldhuizen and Gary B. Lamont, Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.
15. E. Zitzler, K. Deb, and L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, *Evolutionary Computing* **8**(2) (2000) 173–195.
16. E. Zitzler, M. Laumanns, and L. Thiele, Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.