

Exploration of Pareto Frontier Using a Fuzzy Controlled Hybrid Line Search

Crina Grosan and Ajith Abraham

Faculty of Information Technology, Mathematics and Electrical Engineering,
Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence
Norwegian University of Science and Technology, Trondheim, Norway
cgrosan@cs.ubbcluj.ro, ajith.abraham@ieee.org

Abstract

This paper proposes a new approach for multicriteria optimization which aggregates the objective functions and uses a line search method in order to locate an approximate efficient point. Once the first Pareto solution is obtained, a simplified version of the former one is used in the context of Pareto dominance to obtain a set of efficient points, which will assure a thorough distribution of solutions on the Pareto frontier. In the current form, the proposed technique is well suitable for problems having multiple objectives (it is not limited to bi-objective problems) and require the functions to be continuous twice differentiable. In order to assess the effectiveness of this approach, some experiments were performed and compared with two recent well known population-based meta-heuristics ParEGO [11] and NSGA II [2]. When compared to ParEGO and NSGA II, the proposed approach not only assures a better convergence to the Pareto frontier but also illustrates a good distribution of solutions. From a computational point of view, both stages of the line search converge within a short time (average about 150 milliseconds for the first stage and about 20 milliseconds for the second stage). Apart from this, the proposed technique is very simple, easy to implement to solve multiobjective problems.

1. Introduction

The field of multicriteria programming abounds in methods for dealing with different kind of problems. Nevertheless, there is still space for new approaches, which can better deal with some of the difficulties encountered by the existing approaches. There are two main classes of approaches suitable for multiobjective optimization: scalarization methods and nonscalarizing methods. These approaches convert the Multiobjective Optimization Problem (MOP) into a Single Objective Optimization Problem (SOP), a sequence of SOPs, or into another MOP. There are several scalarization methods reported in the literature: weighted sum approach, weighted t -th power approach, weighted quadratic approach, ϵ -constraint approach, elastic constraint approach, Benson approach, etc. are some of them [7]. Since the standard weighted sum encounters some difficulties, several other methods have been proposed to overcome the major drawbacks of this method. These include: Compromise Programming [5], Physical Programming [12][13], Normal Boundary Intersection (NBI) [1], and the Normal Constraint (NC) [14] methods. There is also a huge amount of work reported on population-based metaheuristics for MOP [7] [4] [6][17]. Comprehensive surveys can be found in [18], [10] [15].

In this paper, we propose a new approach which uses a scalarization of the objectives in a way similar to the weighted t -th power approach (where t is 2 and the coefficients values are 1). A line search based technique is used to obtain an efficient solution. Starting with this solution, a set of efficient points are further generated, which are widely distributed along the Pareto frontier using again a line search based method but involving Pareto dominance relationship.

Empirical and graphical results and illustrations obtained by the proposed approach are compared with two well known population based metaheuristics namely ParEGO [11] and NSGA II [2].

The paper is structured as follows: in Section 2 the proposed modified Line Search is presented. Numerical experiments are performed in Section 3. A set of 3 multiobjective optimizations problems are considered. Conclusions and further research plans are presented in Section 4.

2. Line search generator of Pareto frontier

The line search [8] is a standard and well established optimization technique. The standard line search technique is modified in this paper so that it is able to generate the set of non-dominated solutions for a MOP. The approach proposed is called **Line search Generator of Pareto frontier (LGP)** and it comprises of two phases: first, the problem is transformed into a SOP and a solution is found using a line search based approach. This is called as *convergence phase*. Second, a set of Pareto solutions are generated starting with the solution obtained at the end of convergence phase. This is called as *spreading phase*. The *convergence* and *spreading* phases are described below.

Consider the MOP formulated as follows:

Let \mathfrak{R}^m and \mathfrak{R}^n be Euclidean vector spaces referred to as the decision space and the objective space. Let $X \subset \mathfrak{R}^m$ be a feasible set and let f be a vector-valued objective function $f: \mathfrak{R}^m \rightarrow \mathfrak{R}^n$ composed of n real-valued objective functions $f = (f_1, f_2, \dots, f_n)$, where $f_k: \mathfrak{R}^m \rightarrow \mathfrak{R}$, for $k=1,2,\dots, n$. A MOP is given by:

$$\begin{aligned} \min & (f_1(x), f_2(x), \dots, f_n(x)), \\ \text{subject to } & x \in X. \end{aligned}$$

2.1 Convergence phase

The MOP is transformed into a SOP by aggregating the objectives using an approach similar to the weighted t -th

power approach. We consider $t = 2$ and the values of weights equal to 1. The obtained SOP is:

$$\min F = \sum_{i=1}^n f_i^2(x)$$

subject to $x \in X$.

A modified line search method is used to find the optimum of this problem. The modification proposed in this paper for the standard line search technique refers to direction and step setting and also the incorporation of a re-start procedure. To fine tune the performance, the first partial derivatives of the function to optimize are also made use of. The proposed modifications refer to:

- the setting of the direction and step
- the re-starting of the line search method

After a given number of iterations, the process is restarted by reconsidering other arbitrary starting point which is generated by taking into account the result obtained at the end of previous set of iterations.

Direction and step setting

Initially, several experiments were performed in order to set an adequate value for the direction. The standard value +1 or -1 was used and for some functions the value -1 was favourable to obtain good performance. Some experiments were also performed by setting the direction value as being a random number between 0 and 1. It was found that the usage of random number helped to obtain overall very good performance for the entire considered test functions. But usage of the value -1 for direction, obtains almost the same performance similar to that obtained with a random value. So, either of these values (the random one and the value -1) may be used for better performance.

The step is set as follows:

$$\alpha_k = 2 + \frac{3}{2^{2k} + 1} \quad (1)$$

where k refers to the iteration number. The modified *line search* technique is summarized as follows:

Line_search()

Set $k=1$ (Number of iterations)

Repeat

for $i=1$ to No of variables

$p_k = \text{random}; // \text{or } p = -1;$

$$\alpha_k = 2 + \frac{3}{2^{2k} + 1}$$

$$x_i^{k+1} = x_i^k + p_k \cdot \alpha_k$$

endfor

if $F(x^{k+1}) < F(x^k)$ then $x^{k+1} = x^k$.

$k=k+1$

Until $k = \text{Number of iterations}$ (a priori known).

Remarks

- (i) The condition:

$$\text{if } F(x^{k+1}) < F(x^k) \text{ then } x^{k+1} = x^k$$

allows to move to the new generated point only if there is an improvement in the quality of the function.

- (ii) Number of iterations for which line search is applied is apriori known and is usually a small number. For the experiments reported in this paper, the number of these iterations was set to 10.
- (iii) When restarting the line search method (after the insertion of the re-start technique) the value of the iterations number starts again from 1 (this should not be related to the value of α after the first set of iterations (and after each of the following iterations)).

Several experiments were attempted to set a value for the step, starting with random values (until a point is reached for which the objective function achieves a better value); using a starting value for the step and generating random numbers with Gaussian distribution around this number, etc. As a result of the initial experiments performed, it was decided to use equation (1) to compute the step size. But, of course, there are also several other ways to set this.

Incorporation of re-start procedure

In order to restart the algorithm the result obtained in the previous set of iterations (denote it by x) is taken into account and the steps given below are followed:

1. For each dimension i of the point x , the first partial derivative with respect to this dimension is calculated. This means the gradient of the objective function is calculated which is denoted by g . Taking this into account, the bounds of the definition domain for each dimension are recalculated as follows:

$$\text{if } g_i = \frac{\partial F}{\partial x_i} > 0 \text{ then upper bound} = x_i;$$

$$\text{if } g_i = \frac{\partial F}{\partial x_i} < 0 \text{ then lower bound} = x_i;$$
2. The search process is re-started by re-initializing a new arbitrary point between the newly obtained boundaries.

2.2 Spreading phase

At the end of the convergence phase, a solution is obtained. This solution is considered as an efficient (or Pareto) solution. During this phase and taking into account of the existing solution, more efficient solutions are to be generated so as to have a thorough distribution of all several good solutions along the Pareto frontier. In this respect, the line search technique is made use of to generate one solution at the end of each set of iterations. This procedure is applied several times in order to obtain a larger set of non-dominated solutions. The following steps are repeated in order to obtain one non-dominated solution:

Step 1. A set of nondominated solutions found so far is archived. Let us denote it by $NonS$. Initially, this set will have the size one and will only contain the solution obtained at the end of *convergence phase*.

Step2. We apply line search for one solution and one dimension of this solution at one time. For this:

Step 2.1. A random number i between one and $|NonS|$ ($|$ denotes the cardinal) is generated. Denote the corresponding solution by $nonS_i$.

Step 2.2. A random number j between one and the number of dimensions (the number of decision variables) is generated. Denote this by $nonS_{ij}$.

Step 3. Line search is applied for $nonS_{ij}$.

Step 3.1. Set a random value for p between $[-0.5, 1]$.

Step 3.2. Call the fuzzy logic controller to set the value of α (which depends on the problem, on the number of total nondominated solutions which are to be generated, etc.). See Section 2.2.1.

Step 3.3. The new obtained solution new_sol is identical to $nonS_i$ in all dimensions except dimension j which is:

$$new_sol_j = nonS_{ij} + \alpha \cdot p$$

Step 3.4. if $(new_sol_j > upper\ bound)$ or $(new_sol_j < lower\ bound)$

$$then\ new_sol_j = lower\ bound + random \cdot (upper\ bound - lower\ bound).$$

Step 4. if $F(new_sol) > F(nonS_1)$

then discard new_sol

else if new_sol is nondominated with respect to the set $NonS$

then add new_sol to $NonS$ and increase the size on $NonS$ by 1.

Go to step 2.

Step 4. Stop

These steps are repeated until a set on nondominated solutions of a required size is obtained. In our experiments the size of this set is 100. Note that this procedure it very fast and it takes less than 20 milliseconds to obtain 100 non-dominated solutions.

2.2.1 Estimating the Value of α using a Fuzzy Logic Controller

The performance of the line search algorithm is correlated to directly with its careful selection of α value. The use of fuzzy logic controllers to adapt the α value is useful to improve the performance. An FLC is composed by a knowledge base, that includes the information given by the expert in the form of linguistic control rules, a fuzzification interface, which has the effect of transforming crisp data into fuzzy sets, an inference system, that uses them together with the knowledge base to make inference by means of a reasoning method, and a defuzzification interface, that translates the fuzzy control action thus obtained to a real control action using a defuzzification

method. The generic structure of an FLC is shown in Figure 2.

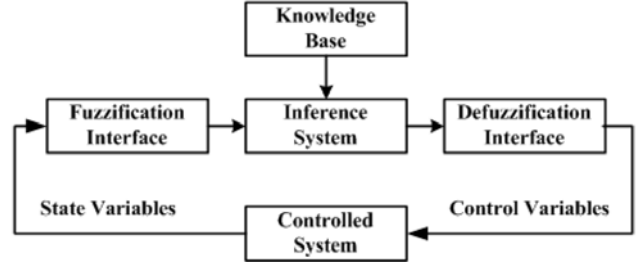


Figure 1. Generic structure of an FLC

In order to set an adequate value for α so that the solutions will have a good distribution on the Pareto front, we are proceeding as follows:

- Select a sample set of solutions uniform distributed on the Pareto front (denoted by SPS) of size equal to the size of the approximation set obtained by the our approach.
- For each point from the approximation set obtained by our approach identify the closest point in SPS.
- Mark each such identified point from SPS.
- Set the value of distribution indices (D_i) as being equal to the number of marked points from SPS.

Our strategy for updating the α value is to consider the changes of the value of maximum distribution indices (D_{im}) and average distribution indices (D_{ia}) in two continuous iterations. The performance may be measured using two error indices.

$$e_1(t) = \frac{D_{im}(t) - D_{ia}(t)}{D_{im}(t)}$$

$$e_2(t) = \frac{D_{ia}(t) - D_{ia}(t-1)}{D_{im}(t)}$$

Where t is time step,

$D_{im}(t)$ is the maximum distribution index at iteration t ,

$D_{ia}(t)$ is the average distribution index at iteration t ,

$D_{ia}(t-1)$ is the average distribution index at iteration $(t-1)$.

A two-dimension FLC system is used, in which there are two parameters e_1 and e_2 . The membership functions are shown in Figure 2, where NL is Negative large, NS is Negative small, ZE is Zero, PS is Positive small, PL is Positive large.

For the controlling the performance, the output $\alpha(t)$ of the fuzzy logic controller is translated using fuzzy *if-then* rules as illustrated in Table 1. Center of gravity is used as defuzzification method. Then we use the crisp value to modify the parameters α as follows:

$$\alpha(t) = \alpha(t-1) + \Delta\alpha$$

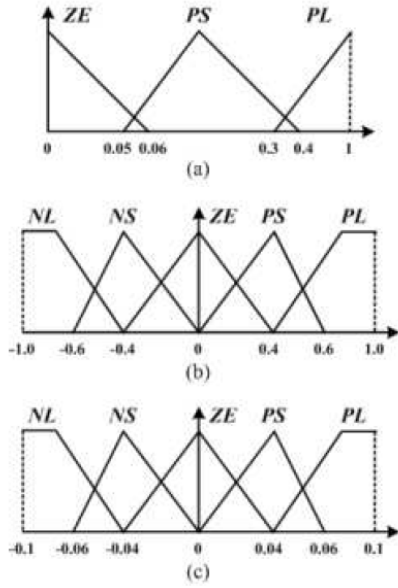


Figure 2. Membership functions. (a) for e_1 , (b) e_2 , (c) for $\Delta\alpha$

		e_2				
		NL	NS	ZE	PS	PL
e_1	PL	PS	PS	ZE	ZE	NS
	PS	PS	ZE	NS	ZE	NS
	ZE	PL	ZE	PS	PS	NL

Table 1. Fuzzy rules for $\Delta\alpha$

For applying the procedure described above, the Pareto front it is supposed to be known (and this is the case in all our experiments considered). In Figure 3, two approximation sets A and B and a sample set of Pareto points (SPS) of size 10 are considered. The value of D_i for the set A is 6 (which means 6 solutions from the SPS are marked) while the value of D_i for the set B is 10. This means set B is obtaining a better distribution on the Pareto front than the set A.

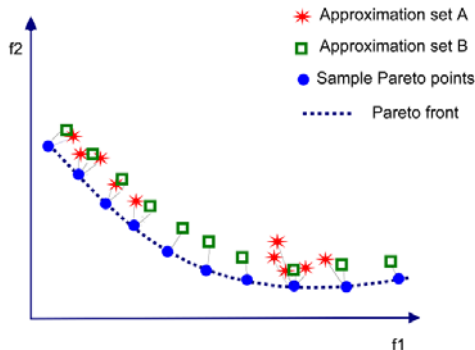


Figure 3. Illustration of Pareto approximation

3. Experiments and Comparisons

In order to assess the performance of LGP, some experiments were performed using some well known bi-objective and three-objective test functions, which are

adapted from [3], [9]. These test functions were also used by the authors of ParEGO [11] and NSGA II [2], which are well known in the computational intelligence community as very efficient techniques for multiobjective optimization. Details about implementation of these two techniques may be obtained from [2] and [11]. Parameters used by ParEGO and NSGA II (given in Table 2) and the results obtained by these two techniques are adapted from [11]. A set of 100 non-dominated solutions obtained by LGP, ParEGO, NSGA II is compared in terms of dominance and convergence to the Pareto set. For the first comparison, two indices were computed for each set of two comparisons: number of solution obtained by the first technique which dominate solutions obtained by the second technique and number of solutions obtained by the first technique which are dominated by the solutions obtained by the second technique. For two sets of A and B of solutions, which are compared, indices are denoted by $Dominates(A, B)$ and $Dominated(A, B)$ respectively. Visualization plots are used to illustrate the distribution of solutions on the Pareto frontier. LGP uses only three parameters:

- number of re-starts: 20 (10 for KNO1);
- number of iteration per each re-start: 10;
- α for the spreading phase (which is set independent for each test function and determined as per Table 1).

Test function KNO1

This test function has two variables and two objectives. It is given by:

$$\text{minimize } f_1 = 20 - r \cdot \cos(\phi)$$

$$\text{minimize } f_2 = 20 - r \cdot \sin(\phi)$$

where

$$r = 9 - \left(3 \sin\left(\frac{5(x_1 + x_2)^2}{2}\right) + 3 \sin(4(x_1 + x_2)) + 5 \sin(2(x_1 + x_2)) + 2 \right)$$

$$\phi = \frac{\pi(x_1 - x_2 + 3)}{12}$$

The distance from the Pareto front is controlled by r and is a function of the sum of the decision variables. The location transverse to the Pareto front is controlled by the difference between the decision variables. Pareto set consists of all pairs whose sum is 4.4116. There are 15 local Pareto fronts and the true Pareto front lies just beyond a local Pareto front which has a larger basin of attraction. The convergence to the Pareto frontier and the distribution of solutions obtained by LGP, ParEGO and NSGA II for the test function DTLZ1a is depicted in Figure 4. Different sizes of the objective space are illustrated in order to incorporate all solutions obtained by all techniques. The behavior of the merit function during the 10 re-starts is depicted in Figure 5.

From the results presented in Table 3 it can be observed that 7 of the solutions obtained by LGP are dominated by solutions obtained by ParEGO and 2 are dominated by solutions obtained by NSGA II. Solutions obtained by LGP dominate all 100 solutions obtained by both ParEGO and NSGA II. 59 of the solutions obtained by NSGA II are dominated by solutions obtained by ParEGO while 42 of the solutions obtained by ParEGO are dominated by solutions obtained by NSGA II.

ParEGO		NSGA II	
Parameter	Value	Parameter	Value
Initial population in latin hypercube	$11d - 1$	Population size	20
Total maximum evaluations	250	Maximum generations	13
Number of scalarizing vectors	11 for 2 objectives 15 for 3 objectives	Crossover probability	0.9
Scalarizing function	Augmented Tchebycheff	Real value mutation probability	$1/d$
Internal genetic algorithm evaluations per iteration	200,000	Real value SBX parameter	10
Crossover probability	0.2	Real value mutation parameter	50
Real value mutation probability	$1/d$		
Real value SBX parameter	10		
Real value mutation parameter	50		

Table 2. Parameters used by ParEGO and NSGA II.

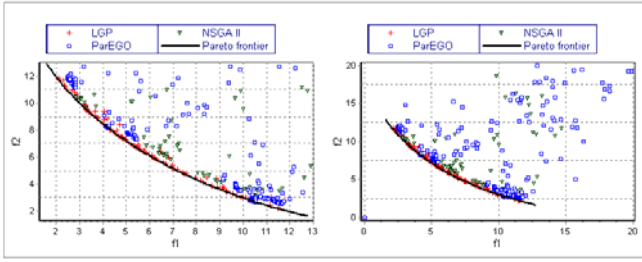


Figure 4. Distribution of solutions on the Pareto frontier obtained by different methods for OKA1

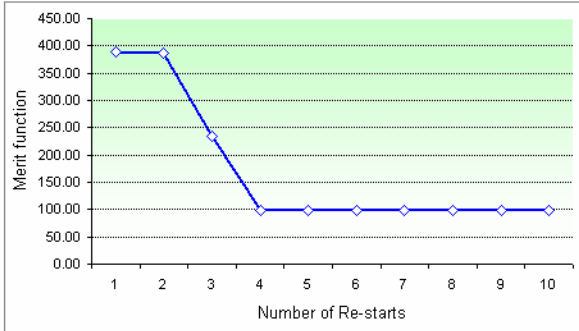


Figure 5. Behavior of merit function for test function KNO1 during the convergence phase

Test function OKA1

This test function [15] is a bi-objective test function having two variables and is defined as:

$$\text{minimize } f_1 = x'_1$$

$$\text{minimize } f_2 = \sqrt{2\pi} - \sqrt{|x'_1|} + 2|x'_2 - 3\cos(x'_1) - 3|^{1/3}$$

where

$$x'_1 = \cos\left(\frac{\pi}{12}\right)x_1 - \sin\left(\frac{\pi}{12}\right)x_2$$

$$x'_2 = \sin\left(\frac{\pi}{12}\right)x_1 + \cos\left(\frac{\pi}{12}\right)x_2$$

$$x_1 \in \left[6 \sin\left(\frac{\pi}{12}\right), 6 \sin\left(\frac{\pi}{12}\right) + 2\pi \cdot \cos\left(\frac{\pi}{12}\right) \right]$$

$$x_2 \in \left[-2\pi \cdot \sin\left(\frac{\pi}{12}\right), 6 \cos\left(\frac{\pi}{12}\right) \right]$$

The Pareto optimal set lies on the curve $x'_2 = 3\cos(x'_1) + 2$, $x'_1 \in [0, 2\pi]$. The solutions obtained by LGP, ParEGO and NSGA II for the test function DTLZ1a are depicted in Figure 6. Different sizes of the objective space are illustrated in order to incorporate all solutions obtained by all techniques. The behavior of the merit function during the 20 re-starts is depicted in Figure 7. From the results presented in Table 4 it can be observed that none of the solutions obtained by LGP are dominated by solutions obtained by either ParEGO or NSGA II. Solutions obtained by LGP dominate 83 solutions obtained by both ParEGO and 64 solutions obtained by NSGA II. 77 of the solutions obtained by NSGA II are dominated by solutions obtained by ParEGO while 59 of the solutions obtained by ParEGO are dominated by solutions obtained by NSGA II.

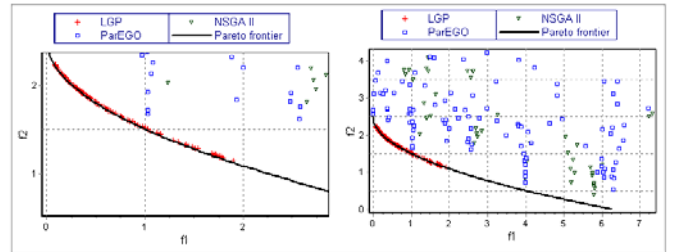


Figure 6. Distribution of solutions on the Pareto frontier for test function OKA1

<i>Dominate</i>	ParEGO	NSGA II	<i>Dominate</i>	LGP	NSGA II	<i>Dominate</i>	LGP	ParEGO
LGP	100	100	ParEGO	7	59	NSGA II	2	42
<i>Dominated</i>	ParEGO	NSGA II	<i>Dominated</i>	LGP	NSGA II	<i>Dominated</i>	LGP	ParEGO
LGP	7	2	ParEGO	100	42	NSGA II	100	59

Table 3. The dominance between solutions obtained by LGP, ParEGO and NSGA II for KNO1.

<i>Dominate</i>	ParEGO	NSGA II	<i>Dominate</i>	LGP	NSGA II	<i>Dominate</i>	LGP	ParEGO
LGP	83	64	ParEGO	0	77	NSGA II	0	59
<i>Dominated</i>	ParEGO	NSGA II	<i>Dominated</i>	LGP	NSGA II	<i>Dominated</i>	LGP	ParEGO
LGP	0	0	ParEGO	83	59	NSGA II	64	77

Table 4. The dominance between solutions obtained by LGP, ParEGO and NSGA II for OKA1.

<i>Dominate</i>	ParEGO	NSGA II	<i>Dominate</i>	LGP	NSGA II	<i>Dominate</i>	LGP	ParEGO
LGP	100	100	ParEGO	0	98	NSGA II	0	60
<i>Dominated</i>	ParEGO	NSGA II	<i>Dominated</i>	LGP	NSGA II	<i>Dominated</i>	LGP	ParEGO
LGP	0	0	ParEGO	100	60	NSGA II	100	98

Table 5. The dominance between solutions obtained by LGP, ParEGO and NSGA II for DTLZ4a.

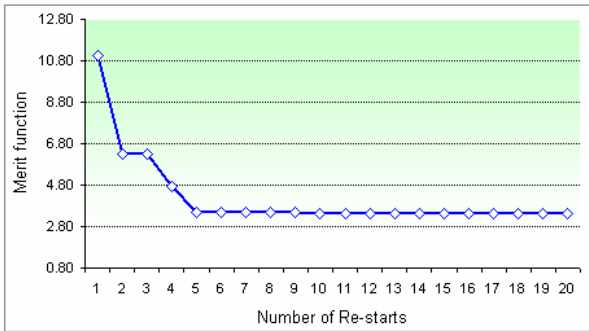


Figure 7. Behavior of merit function for test function OKA1 during the convergence phase.

Test function DTLZ4a

Test function DTLZ4a has three objective functions and 8 decision variables and is given by:

$$\text{minimize } f_1 = (1 + g) \cos\left(\frac{x_1^{100} \pi}{2}\right) \cos\left(\frac{x_2^{100} \pi}{2}\right)$$

$$\text{minimize } f_2 = (1 + g) \cos\left(\frac{x_1^{100} \pi}{2}\right) \sin\left(\frac{x_2^{100} \pi}{2}\right)$$

$$\text{minimize } f_3 = (1 + g) \sin\left(\frac{x_1^{100} \pi}{2}\right)$$

$$g = \sum_{i=3}^8 (x_i - 0.5)^2$$

$$x_i \in [0, 1], i=1, \dots, n, n=8.$$

The Pareto front is 1/8 of the unit sphere centered in origin. The Pareto optimal set consist of all solutions but the first

two decision variables are equal to 0.5 and the first two decision variables may take any value between 0 and 1.

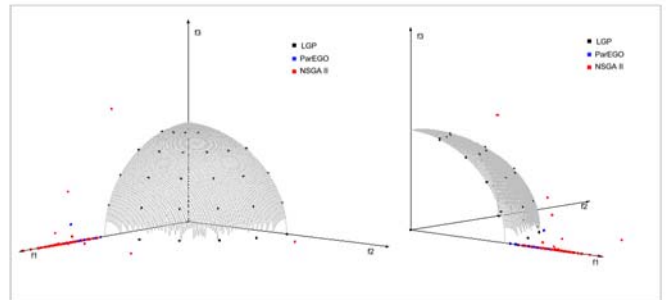


Figure 8. Convergence to the Pareto frontier and distribution of solutions for DTLZ4a

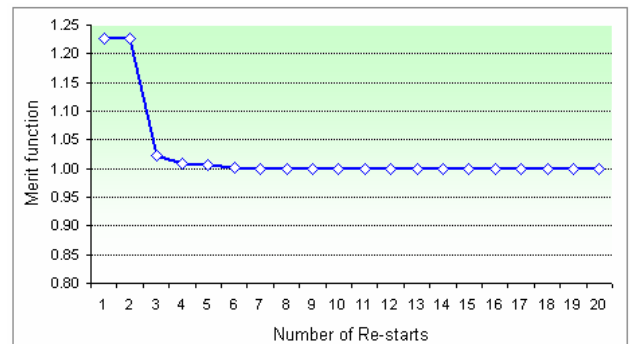


Figure 9. Behavior of merit function for test function DTLZ4 during the convergence phase

The distribution of solutions on the Pareto frontier and the convergence to the Pareto frontier for all the three algorithms is depicted in Figure 8. The convergence of the merit function is depicted in Figure 9. From Figure 8, it can be observed that, compared to ParEGO and NSGA II, LGP

is assuring a very good convergence. The latter two approaches are not converging very well with the parameters used.

As evident from Table 5 none of the solutions obtained by LGP are dominated neither by ParEGO or by NSGA II while solutions obtained by LGP dominate all 100 solutions obtained by ParEGO and NSGA II. 98 of the solutions obtained by NSGA II are dominated by solutions obtained by ParEGO while 60 of the solutions obtained by ParEGO are dominated by solutions obtained by NSGA II.

4. Conclusions

The paper proposes a new approach for multiobjective optimization which uses an aggregation of objectives and transforms the MOP into a SOP. A line search based technique is applied in order to obtain one solution. Starting from this solution a simplified version of the initial line search is used in order to generate solutions with a well distribution on the Pareto frontier. Numerical experiments performed show that the proposed approach is able to converge very fast and provide a very good distribution (even for discontinuous Pareto frontier) while compared with state of the art population based metaheuristics such as ParEGO and NSGA II. Compared to NSGA II and ParEGO, LGP has only few parameters to adjust. It is computationally inexpensive, taking less than 200 milliseconds to generate a set of nondominated solutions well distributed on the Pareto frontier.

The only inconvenience is that LGP involves first partial derivatives, which makes it be restricted to a class of problems which are continuous twice differentiable. But almost all practical engineering design problems are continuous differentiable.

References

- [1] Das, I. and Dennis, J., A Closer Look at Drawbacks of Minimizing Weighted Sums of Objective for Pareto Set Generation in Multicriteria Optimization Problems, *Structural Optimization*, 14, pp. 63–69, 1997.
- [2] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transaction on Evolutionary Computation*, 6(2), 181-197, 2002.
- [3] Deb, K., Thiele, L., Laumanns, M. and Zitzler, E., Scalable multi-objective optimization test problems. *Proceedings of the Congress on Evolutionary Computation (CEC-2002)*. (Honolulu, USA). pp. 825-830, 2002.
- [4] Deb, K., *Multiobjective Optimization Using Evolutionary Algorithms*, Series in Systems and Optimization, John Wiley and Sons, Chichester, England, 2001.
- [5] Chen, W., Wiecek, M. M., and Zhang, J., Quality Utility – A Compromise Programming Approach to Robust Design, *Journal of Mechanical Design*, 121, pp. 179–187, 1999.
- [6] Ehrgott, M., and Gandibleux, X., *Multiobjective Combinatorial Optimization, Multiple-Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, Kluwer Academic Publishers, Boston, Massachusetts, 2002.
- [7] Ehrgott, M., and Wiecek, M., *Multiobjective Programming*, In *Multiple Criteria Decision Analysis: State of the art surveys*, J. Figueira et al. (Eds.), International Series in Operations Research & Management Science, Volume 78, Springer New York, 2005.
- [8] Gould, N., *An introduction to algorithms for continuous optimization*, Oxford University Computing Laboratory Notes, 2006.
- [9] Huband, S., Hingston, P., Barone, L., and While, L., A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit, *IEEE Transactions on Evolutionary Computation*, 10(5), pp. 477-506, 2006.
- [10] Jahn, J., *Vector Optimization: Theory, Applications, and Extensions*, Springer-Verlag, Berlin, 2004.
- [11] Knowles, J., ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10 (1), 50-66, 2006.
- [12] Messac, A. and Mattson, C. A., Generating Well-Distributed Sets of Pareto Points for Engineering Design using Physical Programming, *Optimization and Engineering*, 3, pp. 431–450, 2002.
- [13] Messac, A., Ismail-Yahaya, A. *Multiobjective Robust Design using Physical Programming, Structural and Multidisciplinary Optimization*, *Journal of the International Society of Structural and Multidisciplinary Optimization (ISSMO)*, Springer, 23(5), pp. 357-371, 2002.
- [14] Messac, A., Ismail-Yahaya, A., and Mattson, C. A., The Normalized Normal Constraint Method for Generating the Pareto Frontier, *Structural and Multidisciplinary Optimization*, 25(2), pp. 86–98, 2003.
- [15] K.M. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Norwell, 1999.
- [16] Okabe, T., Jin, Y., Olhofer, M., and Sendhoff, B., On test functions for evolutionary Multiobjective optimization, In *Proceedings of 8th Conference on Parallel Problems Solving from Nature*, Springer Verlag Germany, pp. 792-802, 2004.
- [17] Resende MGC, de Sousa JP (Eds), *Metaheuristics: Computer Decision-Making*, Kluwer Academic Publishers, The Netherlands, 2004.
- [18] S. Ruzika, and M. M. Wiecek, Approximation Methods in Multiobjective Programming, *Journal of Optimization Theory and Applications: Vol. 126, No. 3*, pp. 473–501, 2005
- [19] Veldhuizen, D.A.V. Lamont, G.B., *Multiobjective Evolutionary Algorithm Test Suites*, In *Proceedings of ACM Symposium on Applied Computing*, J. Carroll et al. (Eds.), San Antonio, Texas, pp. 351-357, 1999.