Multiobjective Particle Swarm Optimization Using Fuzzy Logic

Yazdani, Hossein ; Kwasnicka, Halina; Ortiz-Arroyo, Daniel

Published in:
Lecture Notes in Computer Science

DOI (link to publication from Publisher):
10.1007/978-3-642-23935-9_22

Publication date:
2011

Document Version
Early version, also known as pre-print

Multiobjective Particle Swarm Optimization Using Fuzzy Logic

Hossein Yazdani[1], Halina Kwasnicka[1], and Daniel Ortiz-Arroyo[2]

[1] Institute of Informatics, Wroclaw University of Technology, Wroclaw, Poland
[2] Electronics Department, Computational Intelligence and Security Laboratory, Aalborg University, Denmark
hyazda10@student.aau.dk, halina.kwasnicka@pwr.wroc.pl, do@es.aau.dk

Abstract. The paper presents FMOPSO a multiobjective optimization method that uses a Particle Swarm Optimization algorithm enhanced with a Fuzzy Logic-based controller. Our implementation makes use of a number of fuzzy rules as well as dynamic membership functions to evaluate search spaces at each iteration. The method works based on Pareto dominance and was tested using standard benchmark data sets. Our results show that the proposed method is competitive with other approaches reported in the literature.

Keywords: Particle Swarm Optimization, Fuzzy Logic, multiobjective optimization.

1 Introduction

Optimization modeling is one of the most powerful techniques to find optimal solutions of problems in application areas such as economy, industry, finance, and others. For instance in economic applications, profit and sales must be maximized and cost should be as low as possible [5]. The goal is to find the best solution x^* from a set of possible solutions X according to a set of criteria $F = \{f_1, f_2, ..., f_n\}$. This set of criteria is expressed as a mathematical function named objective function [5].

General optimization (GO) methods are categorized into two main classes: deterministic and probabilistic methods. The deterministic methods work based on heuristics, such as adding punishment to escape from local minima. Heuristics use the information currently gathered by the algorithm to help deciding which solution candidate should be tested next or how the next solution can be produced [5]. Probabilistic methods estimate probabilities to decide whether the search should depart from the neighbourhood of a local minimum.

Particle Swarm Optimization is one of the methods that have been proposed to solve GO problems [6].

Multiobjective optimization algorithms are designed to optimize a set of objective functions. The simplest methods rely on optimizing the weighted sum $g(x)$ of all functions (criteria) $f_i(x) \in F$ [5]. The mathematical foundations for multiobjective optimization that considers conflicting criteria in a fair way was

P. Jędrzejowicz et al. (Eds.): ICCCI 2011, Part I, LNCS 6922, pp. 224–233, 2011.

laid by Vilfredo Pareto 110 years ago [5]. Pareto optimization is based on the definition of domination: a solution x_1 dominates (is preferred to) solution x_2 ($(x_1 \vdash x_2)$) if x_1 is better than x_2 in at least one objective function and not worse with respect to all other objectives.

A solution $x^* \in X$ is Pareto optimal (belongs to the optimal set X^*) if it is not dominated by any other solution in the problem space X. In terms of Pareto optimization, X^* is called the Pareto optimal set, denoted as P^* [6].
The set $PF^* = \{f_1(x), f_2(x), \cdots, f_k(x) \mid x \in P^*$ is called Pareto Front PF^*

Particle Swarm Optimization (PSO) was introduced by R.C. Eberhart and J. Kennedy in 1995 [6]. PSO is an adaptive, global optimization method which is based on updating the value of each particle to obtain the best solution. In this method, each potential solution is called a *particle*, and each particle has a *fitness value* which is calculated by a fitness function. This fitness value is the one that should be optimized. PSO works by generating some random solutions consisting of particles, where each particle is iteratively updated with two different values. One is the best value reached by the particle so far (called *local best* or *lbest*), and the second one is the best value obtained by any particle so far (called *global best* or *gbest*) [7]. Based on these two best values, the velocity of particle p and its position are updated and then the fitness of the particle is calculated. In this way an optimal solution may be found after some number of iterations. There are two different general PSO models, one is called *local version* and another is *global version* [6]. In the local version, each particle flies through the search space to adjust the velocity according to its best achieved performance so far and the best performance achieved by the neighbourhood particles. In the global version, the particle's velocity is adjusted according to its best achieved performance so far and the best performance achieved by all particles [2].

The main idea of the algorithm presented in this paper is to break down the problem to be solved into several simpler ones, and evolve particles to find Pareto Fronts in smaller spaces. A combination of these fronts constitutes the final Pareto Front. Fuzzy logic is used in deciding which part of the problem should be selected for the next iteration. The objective is to use that part, on which finding the non-dominated particles is more probable.

This paper is organized in the following way. In section 2 some related work is briefly described. The proposed approach is presented in section 3. The experimental study of the method is described in section 4. The last section summarizes the paper and presents some conclusions.

2 Related Work

PSO has become a popular optimization method that is widely studied and compared to other approaches. Given that the literature on this subject is extensive, in this section we present a brief summary of related work.

[3] describes the use of PSO algorithm as a tool to optimize path finding. Authors use the Mamdani inference fuzzy system to decide when to increase or decrease the velocity of particles and in which direction the positions of particles should be changed.

A new algorithm was introduced in [1] where the search space is divided according to the best achieved values accumulated in a repository. The algorithm works based on a hypercube. Then from the chosen hypercube (on the basis of fitness of all hypercubes) one particle is selected randomly.

POS is combined with Fuzzy Logic in [4], where Fuzzy Logic helps the particle to improve an existing solution by replacing the low quality links with high quality links. The local version of PSO model is used to optimize topology design of distributed local area networks.

PSO and multiobjective optimization (MO) in n-dimensional search space are discussed in [6]. Authors discuss weighted aggregation approaches such as: Dynamic weighted aggregation (DWA), Conventional weighted aggregation (CWA), Bang-Bang Weighted Aggregation (BWA), Vector Evaluated Genetic Algorithm (VEGA) and Vector Evaluated Particle Swarm Optimization (VEPSO). Authors introduced a maximum value for velocity to improve the performance of the basic PSO by avoiding high increases in velocity values. Another topic discussed in that paper is the range of values for the basic parameters c_1,c_2 (connected with the influence of local and global best solution).

In [8], a new algorithm for two-objective functions in two-dimensional fitness value space is introduced. Authors assume fixed fitness values of the first objective function, and try to optimize the second objective function.

Concepts of PSO, neighbourhood topology, multi-objective optimization (MOO) and leaders in MOO are discussed in [12]. Authors explain how to select the leader from all non-dominated solutions as a global best solution to guide the algorithm to get the new particles. Nearest neighbor density estimator and kernel density estimator are some approaches mentioned in this paper.

3 Multiobjective Particle Swarm Optimization Using Fuzzy Logic (FMOPSO)

The FMOPSO method is inspired by the "Divide and conquer" or "Sub population" approach. The main idea of FMOPSO is to break the original problem down to several parts with less complexity. Then by gathering particles (solutions) from these smaller problems, a better overall solution could be obtained. The search space is divided into an arbitrary number of major spaces (we use five major spaces), further, every major space is divided into an arbitrary number of minor spaces (in our experiments it is also five). To obtain better results, we combined population and pareto-based approaches. Our algorithm determines which area has more priority for being selected as a new search space for next iteration. For this purpose we make use of a fuzzy controller that evaluates search spaces at each iteration. In the calculation of velocity and position of a new particle we use the basic PSO algorithm:

$$v_{(i+1)} = v_i + c_1 * r_1 * (lbest_i - p_i) + c_2 * r_2 * (gbest_i - p_i)$$
$$p_{(i+1)} = p_i + v_{(i+1)} \quad (1)$$

where $c_1 = 1.4,\ c_2 = 1.5,\ r_1\ , r_2$ are random value in $[0, 1]$. Half a range of the variable is the limit for maximal velocity. When a particle's velocity is near to zero, the particle is moved to a different dimension in the n-dimensional search space. This allows particles to escape from the local optimum. Additionally, we use two accumulators, named as *repository* and *deleted-repository*, thanks to it the fuzzy controller can check densities of both – non-dominated and dominated particles. The fuzzy controller decides which search space should be selected to choose the *gbest*.

The main steps of FMOPSO. The FMOPSO algorithm is briefly presented in Algorithm 1. The first step, *Initialize parameters*, consists of four methods, *Get ranges*, *Initialize particles*, *Initialize fitness*, and *Initialize parts*, i.e., major and minor parts.

In *Compute velocity* we perform three steps, each has a different strategy for choosing *gbest*: (1) select *gbest* from all *gbest* achieved so far (if we have many candidates, one is randomly selected), (2) fuzzy controller looks for the best search space to get *gbest* with respect to the density of particles in both repository and deleted-repository, (3) look for the *gbest* based on the major part. *Evaluate fitness* evaluates the particle's fitness value (the objective functions). *Update Lbest* method updates best position achieved by the considered particle so far. *Evaluate Non-dominated* method is used to choose the non-dominated particle.

Algorithm 1. FMOPSO general algorithm

```
Initialize parameters();
Update Lbest();
Evaluate Non-dominated();
while i <= max.iteration do
   Compute Velocity();
   Evaluate fitness();
   Update Lbest();
   Evaluate Non-dominated();
   i = i + 1;
end while
Make report();
```

Fuzzy Logic Controller. Whole search space is divided into five smaller parts, and every smaller search subspace is divided into five subsections. The task is to decide which subsection should be selected for next iteration. This is done on the basis of evaluating the availability of non-dominated particles in a search space. Fuzzy controller checks the density of particles in both the repository and deleted-repository. The interesting area is where the density of dominated particle is low or medium and the density of non-dominated particle is medium and high. This is the reason why our fuzzy controller looks at densities in both repositories.

Membership Functions for Input and Output Variables. Our fuzzy controller uses three fuzzy sets for each of the input variables, *Density* and *No. of*

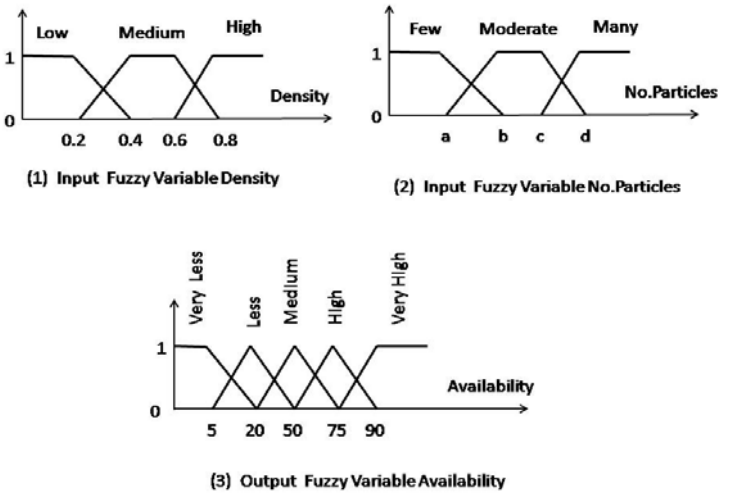

Fig. 1. Graphical representation of membership function

Particles, for both accumulators – *repository* and *deleted-repository*. For variable *Density* they are: *Low* (L), *Medium* (M), and *High* (H). For *No. Particles*: *Few* (F), *Moderate* (MO), and *Many* (MA).

Five fuzzy sets are defined for output variable *Availability* of non-dominated particles: *Very High Availability* (VHP), *High Availability* (HP), *Medium Availability* (MP), *Less Availability* (LSP), *Very Less Availability* (VLSP).

Membership functions are presented in Fig. 1. It is seen in this figure, that the boundaries of fuzzy sets for variable No. of Particles change with increasing iterations. In that figure the values of points $a,\ b,\ c,\ d$ in the fuzzy sets are calculated using a coefficient $coefficient = 0.1 * population.size * current.iteration$, in the following way:

$a = coefficient$,
$b = coefficient + coefficient/2$,
$c = 2 * coefficient + coefficient/2$,
$d = 3 * coefficient$.

Fuzzy Rule Base. We have defined eighteen different fuzzy rules for the controller. They are shown in Table 1. The detailed rules are:

1. If *density* in dominated set is low or medium and *density* in non-dominated set is low and *NO.Particles* is few, *availability* is high
2. If *density* in dominated set is low or medium and *density* in non-dominated set is low and *NO.Particles* is moderate, *availability* is middle
3. If *density* in dominated set is low or medium and *density* in non-dominated set is low and *NO.Particles* is many, *availability* is less
4. If *density* in dominated set is high and *density* in non-dominated set is low and *NO.Particles* is few, *availability* is middle

Table 1. 3-dimensional matrix defining the fuzzy rules: availability depending on densities in the both accumulators

Repository →	L			M			H		
Del-rep↓ No.P →	F	MO	MA	F	MO	MA	F	MO	MA
L	HP	MP	LSP	HP	HP	VHP	HP	VHP	VHP
M	HP	MP	LSP	MP	MP	MP	MP	HP	HP
H	MP	LSP	VLSP	MP	LS	VLSP	MP	MP	LSP

5. If *density* in dominated set is high and *density* in non-dominated set is low and *NO.Particles* is moderate, *availability* is less
6. If *density* in dominated set is high and *density* in non-dominated set is low and *NO.Particles* is moderate, *availability* is very Less
7. If *density* in dominated set is low and *density* in non-dominated set is medium and *NO.Particles* is few or moderate, *availability* is very high
8. If *density* in dominated set is low and *density* in non-dominated set is medium and *NO.Particles* is many, *availability* is very high
9. If *density* in dominated set is medium and *density* in non-dominated set is medium and *NO.Particles* is few or moderate or much, *availability* is middle
10. If *density* in dominated set is high and *density* in non-dominated set is medium and *NO.Particles* is few, *availability* is middle
11. If *density* in dominated set is high and *density* in non-dominated set is medium and *NO.Particles* is moderate, *availability* is less
12. If *density* in dominated set is high and *density* in non-dominated set is medium and *NO.Particles* is many, *availability* is very less
13. If *density* in dominated set is low and *density* in non-dominated set is high and *NO.Particles* is few, *availability* is high
14. If *density* in dominated set is low and *density* in non-dominated set is high and *NO.Particles* is moderate or many, *availability* is very high
15. If *density* in dominated set is medium and *density* in non-dominated set is high and *NO.Particles* is few, *availability* is middle
16. If *density* in dominated set is medium and *density* in non-dominated set is high and *NO.Particles* is moderate or many, *availability* is high
17. If *density* in dominated set is high and *density* in non-dominated set is high and *NO.Particles* is few or moderate, *availability* is very middle
18. If *density* in dominated set is high and *density* in non-dominated set is high and *NO.Particles* is many, *availability* is less

The fuzzy controller fires the applicable rules and calculates the fuzzy output using Mamdani (max-min) implication technique. To get the crisp value from all fired rules, the weighted-average defuzzification technique is used.

4 Experimental Results

To verify the performance of our method, FMOPSO was run using different benchmark data sets and different values of parameters: five test functions (ZDT test set), 100 iterations, and for five different numbers of particles $(5, 25, 50, 75, 100)$.

Table 2. ZDT two-objectives problems

Name	Problem	Type	Parameter Domains
ZDT1	$f_1(x) = x_1$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$	Convex	[0,1]
ZDT2	$f_1(x) = x_1$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $h(f_1, g) = 1 - (\frac{f_1}{g})^2$	Non-Convex	[0,1]
ZDT3	$f_1(x) = x_1$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g})\sin(10\Pi f_1)$	Non-Convex, Disconnected	[0,1]
ZDT4	$f_1(x) = x_1$ $g(x) = 1+10(n-1)+\sum i = 2^n(x_i^2-10\cos(4\Pi x_i))$ $h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$	Convex, Multi-Modal	[0,1]
ZDT6	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\Pi x_1)$ $g(x) = 1 + 9(\frac{\sum_{i=2}^{10} x_i}{9})^{0.25}$ $h(f_1, g) = 1 - (\frac{f_1}{g})^2$	Non-Convex, Non-uniformally Spaced	[0,1]

Zitzler-Deb-Thiele (ZDT) set contains scalable problems according to the number of decision variables [9,10,11]. ZDT problems contain two objective problems. Given f_1, the second criterion is a composite function $f_2(x) = g(x)h(f_1(x), g(x))$, both objectives should be minimized. Table 2 contains defined 2-dimensional problems. Three performance metrics were used: Generational distance, Spacing and Error ratio.

Generational Distance (GD). GD was introduced by Van Veldhuizen and Lamont [12] to calculate the distance between particles in non-dominated set generated by the method and the Pareto Optimal set.

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}, \quad d_i = min_{k=1}^{|p*|}\sqrt{\sum_{m=1}^{M}(f_m^i - f_m^{*(k)})^2} \tag{2}$$

where n is the number of particles in the non-dominated set and d_i is the Euclidean distance between particle in the non-dominated set and the closest one from Pareto Optimal set and $f_m^{*(k)}$ is the m-th objective value of the k-th member of p^*. It is obvious that the smaller values of GD are preferred. GD close to *zero* indicates that non-dominated set is a part of Pareto Front.

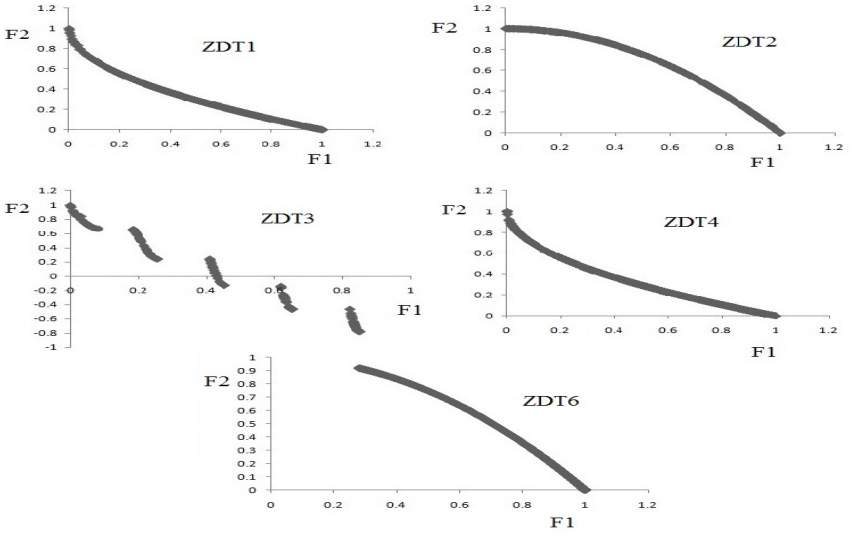

Fig. 2. Pareto front produced by FMOPSO for the ZDT set test problem

Spacing (SP). SP was introduced by Schott(1995) [12] to calculate distance between consecutive solutions from non-dominated set.

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \bar{d})^2} \quad \bar{d} = \sum_{i=1}^{n}\frac{d_i}{n} \tag{3}$$

The smaller spacing values are better, they are the standard deviations of different d_i values.

Error Ratio (ER). ER was introduced by Van Velshuizen [12] to measure the number of non-dominated particles which are not the member of Pareto Optimal set.

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{4}$$

where $e_i = 1$ if i is not a member of Pareto Optimal set and $e_i = 0$ otherwise. $ER = 0$ indicates that all solutions in non-dominated set are members of Pareto Optimal set. Figure 2 presents the results produced by our method.

Table 3 shows the result of FMOPSO compared to common Evolutionary Algorithms: NSGA-II real coded, NSGA-II binary coded, SPEA and PAES. In this table the results displayed correspond to the mean and variance with respect to the convergence metric.

Table 4 shows the result achieved by FMOPSO and other recently presented MOPSO algorithms: OMPSO, SMPSO, MOPSO-TVAC, MOPSO-TVIW. In this table the results displayed correspond to the median and IQR indicator with respect to the Delta metric.

Table 3. Convergence metric

Algorithm		ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	$\bar{\gamma}$	0.033482	0.072391	0.114500	0.513053	0.296564
Real-coded	δ_γ^2	0.004750	0.031689	0.007940	0.118460	0.013135
NSGA-II	$\bar{\gamma}$	0.000894	0.000824	0.043411	3.227636	7.806798
Binary-coded	δ_γ^2	0	0	0.000042	7.307630	0.001667
SPEA	$\bar{\gamma}$	0.001249	0.003043	0.044212	9.513615	0.020166
	δ_γ^2	0	0.000020	0.000019	11.321067	0.000923
PAES	$\bar{\gamma}$	0.082085	0.126276	0.023872	0.854816	0.085469
	δ_γ^2	0.008679	0.036877	0.00001	0.527238	0.006664
FMOPSO	$\bar{\gamma}$	0.00028	0.00000287	0.00060	0.00004	0.00016
	δ_γ^2	0.00000	0.00000	0.00000	0.00000	0.00000012

Table 4. Delta metric

Algorithm		ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
OMOPSO	$\bar{x}$	$7.98e^{-02}$	$7.46e^{-02}$	$7.13e^{-01}$	$8.69e^{-01}$	$2.90e^{-01}$
	IQR	$1.4e^{-02}$	$1.6e^{-02}$	$1.0e^{-02}$	$5.9e^{-02}$	$1.1e^{+00}$
SMPSO	$\bar{x}$	$7.66e^{-02}$	$7.33e^{-02}$	$7.10e^{-01}$	$9.81e^{-02}$	$2.83e^{-01}$
	IQR	$1.4e^{-02}$	$1.6e^{-02}$	$7.2e^{-03}$	$1.4e^{-02}$	$1.2e^{+00}$
MOPSO-TVAC	$\bar{x}$	$1.01e^{-01}$	$8.71e^{-01}$	$7.81e^{-01}$	$2.05e^{-01}$	$1.33e^{+00}$
	IQR	$1.3e^{-02}$	$1.3e^{-02}$	$7.1e^{-02}$	$3.6e^{-02}$	$5.7e^{-02}$
MOHPSO	$\bar{x}$	$1.10e^{-01}$	$9.01e^{-02}$	$7.71e^{-01}$	$9.02e^{-01}$	$1.29e^{+00}$
	IQR	$2.7e^{-02}$	$1.9e^{-02}$	$5.9e^{-02}$	$1.6e^{-01}$	$4.3e^{-02}$
MOPSO-TVIW	$\bar{x}$	$8.39e^{-02}$	$7.09e^{-02}$	$7.12e^{-01}$	$1.29e^{-01}$	$1.11e^{+00}$
	IQR	$1.6e^{-02}$	$2.0e^{-02}$	$9.5e^{-03}$	$3.4e^{-01}$	$1.2e^{+00}$
FMOPSO	$\bar{x}$	$1.75e^{-02}$	$1.75e^{-02}$	$4.0e^{-02}$	$1.9e^{-02}$	$1.5e^{-3}$
	IQR	$3.88e^{-02}$	$1.2e^{-03}$	$5.1e^{-03}$	$4.6e^{-04}$	$4.1e^{-04}$

5 Conclusions and Future Work

We have proposed a new algorithm that combines a population based optimization techniques, the Pareto based approach, and a fuzzy controller. The experiments with the ZDT benchmark data sets indicate that FMOPSO approach is able to find the non-dominated particles that are close to the Pareto Front.

Our modification of PSO lies in adding some fuzzy knowledge to make the method more intelligent. Thanks to this knowledge FMOPSO decides where the best area of search space is.

The complexity of the method could be reduced by using sorted balanced tree (AVL or R & B tree) instead of the current array that is being used. The complexity of updating repository is currently $O(kN^2)$ where N is the size of swarm and k is the number of objectives. Complexity of the updating process for all iterations (M) is $O(kMN^2)$ [12], where complexity of insertion into AVL tree is at most $O(log(N + \frac{3}{2} + log(\sqrt{5}) - 3)$ rebalancing operations and $O(log(N +$

$\frac{3}{2} + log(\sqrt{5}) - 4)$ rebalancing operations for deletion, if N is the maximum size of nodes [13]. The use of this kind of tree will improve the time for updating the repository.

The approach used in FMOPSO, namely – finding the most promising areas of search space, i.e., the suitable ranges of particular variables, seems to be good way for multi-objective optimization methods. For example, the best values of variables $x_2, x_3, ..., x_{30}$ from the range [0,1] in ZDT1 data is zero. The system should find the best area for these variables by analyzing them with respect to the given ranges. Searching the promising subspaces at each iteration, on the basis of history analysis, should allow the method to find very good solutions in relatively short time. In future work we plan to improve the fuzzy controller to include a user knowledge base and other fuzzy sets.

References

1. Weise, T.: Global Optimization Algorithms Theory and Application. EBook. IEEE Press, Los Alamitos (2009)
2. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through Particle Swarm Optimization. Natural Computing 1, 235–306 (2002)
3. Clerc, M.: Particle Swarm Optimization. Wiley-ISTE, Chichester (2006)
4. Das, S., Abraham, A., Konar, A.: Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. Studies in Computational Intelligence (SCI), vol. 116, pp. 1–38 (2008)
5. Ghanizadeh, A., Sinaie, S., Abarghouei, A.A., Shamsuddin, S.M.: A fuzzy-particle swarm optimization based algorithm for solving shortest path problem. In: 2nd International Conference on Computer Engineering and Technology, ICCET, pp. V6-404–V6-408 (2010)
6. Coello Coello, C.A., Lechuga, M.S.: MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1051–1056 (2002)
7. Khan, S.A.: Design and analysis of evolutionary and swarm intelligence techniques for topology design of distributed local area networks, ch. 9, University of Pretoria (2009)
8. Hu, X., Eberhart, R.: Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Congress on Evolutionary Computation, vol. 2, pp. 1677–1681 (2002), 0-7803-7282-4/02 IEEE
9. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
10. Coello Coello, C.A., Dhaenens, C., Jourdan, L.: Advances in Multi-Objective Nature Inspired Computing. Springer, Heidelberg (2010)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
12. Reyes-Sierra, M., Coello Coello, C.A.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
13. Larsen, K.S.: AVL trees with relaxed balance. In: Parallel Processing Symposium, pp. 888–893 (1994)