Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011

Hotel Equatorial Bangi-Putrajaya, Malaysia, 14 - 15 January 2011

ISBN 978-983-42366-4-9

Fuzzy Preference Incorporated Evolutionary Algorithm for Multiobjective Optimization

Surafel Luleseged Tilahun[#], Ong Hong Choon[*]

[#]*School of Mathematical Science, Universiti Sains Malaysia*
11800 USM, Penang, Malaysia

[#]*Tel.:+60124113184, E-mail: surafelaau@yahoo.com*

**Tel.:+60164981802, E-mail: hcong@cs.usm.my*

***Abstract*— Multiobjective evolutionary method is a way to overcome the limitation of the classical methods, by finding multiple solutions within a single run of the solution procedure. The aim of having a solution method for multiobjective optimization problem is to help the decision maker in getting the best solution. Usually the decision maker is not interested in a diverse set of Pareto optimal points. So, it is necessary to incorporate the decision maker's preference so that the algorithm gives out alternative solutions around the decision maker's preference. The problem in incorporating the decision maker's preference is that the decision maker may not have a solid guide line in comparing tradeoffs of objectives. However, it is easy for the decision maker to compare in a fuzzy way. This paper discusses on incorporating a fuzzy tradeoffs in the evolutionary algorithm to zoom out the region where the decision maker's preference lies. By using test functions it has shown that it is possible to give points in the region on the Pareto front where the decision maker's interest lies.**

***Keywords*— Multiobjective Optimization, Fuzzy Preference, Evolutionary Algorithm.**

I. Introduction

Most decision making deals with optimizing more than one and usually conflicting objectives. Choosing one from possible set of actions is very difficult due to the conflict of objectives. When one tries to choose an action in favour of one objective he is worsening other competing objective/s. The problem becomes one of ordering or comparing vectors. There is a need to find Pareto solutions; solutions whose outcome is not dominated by any other outcome of the set of possible actions. Selecting the best solution from the given set of Pareto set depends on the preference of the decision maker/s.

Different methods have been developed in dealing with this problem. The older and almost direct forward approach is to change the problem into single objective optimization. One of the most commonly known methods is the weighting method [1]. In this method the decision maker is supposed to give weights for each objective and by multiplying the functions to be optimized by the corresponding weights and taking their sum we end up having a real valued function to optimize. Trade-off method ([2], [3]) is also a method of changing the multiobjective optimization problem into single objective optimization problem, in such a way that all the objective functions except one will be put into a constraint set by putting maximum affordable values for minimization problem. There are many more methods with almost similar fashion; to mention some of them: Benson's Method [1], Utility function method [4], Lexicographic optimality [5]. The above methods will give a single solution and each has its own advantages and limitations. It will be good if we manage to give a set of Pareto optimal solutions for the decision maker to choose among the available finite set of solutions.

Multiobjective evolutionary algorithm is perhaps the best choice for such purpose. An evolutionary algorithm is an algorithm which is inspired by natural evolution proposed by Darwin [6]. In the algorithm a population generated randomly and some members will be chosen for crossover and mutation depending on their fitness. After crossover and mutation a population with better fitness will be constructed by replacing some members of the old population with members of the new population. By performing this step in a finite number of times we end up

having a set of relatively better solutions. Since the use of evolutionary algorithm on multiobjective optimization problems, a lot of studies have been done. But evolutionary algorithms for multiobjective optimization incorporating the decision makers preference has not been explored enough [7]. The studies done on this aspect depend on the importance comparison of each couple of objectives by the decision maker as important, very important and so on ([8], [9]). Even though giving importance order is easier than assigning tradeoffs, there are very sensitive issues in reality that needs an optimal solution under any cost.

This paper discusses on the incorporation of the decision maker's fuzzy preference to the evolutionary analysis, particularly in the genetic algorithm. To do so the decision maker is supposed to give the acceptable fuzzy tradeoff of one objective for a unit increase, if it is maximization (otherwise decrease) for another objective. By combining these input fuzzy tradeoffs we can generate the range of weights of the objective functions with some acceptability. From the acceptability function one can generate random weights for the objective functions using an appropriate probability distribution which expresses the acceptability relationship. And this will be incorporated in the evolutionary algorithm to zoom out the Pareto front of the decision maker's preference.

II. Problem Statement and Preliminaries

A. *Multiobjective Optimization*

A multiobjective optimization problem is an optimization problem with more than one objective functions under a set of feasibility conditions. Multiobjective optimization problems can be found in various fields: product and process design, finance, aircraft design, the oil and gas industry, automobile design, or wherever optimal decisions are needed to be taken in the presence of trade-offs between two or more conflicting objectives. Maximizing profit and minimizing the cost of a product; maximizing performance and minimizing fuel consumption of a vehicle; and minimizing weight while maximizing the strength of a particular component are examples of multiobjective optimization problems. Since one can switch between maximization and minimization problems by multiplying the objective function by -1, we consider a minimization problem.

Mathematically,

Let $F:\Re^n \to \Re^k, F(x)=(f_1(x), f_2(x),..., f_k(x))$ and $S \subseteq \Re^n$

then

$$\min\ F(x) = (f_1(x), f_2(x), \ldots, f_k(x))$$
$$s.t.\ x \in S \subseteq \Re^n$$

is known as a multiobjective optimization problem

A point *x'* in the domain is said to be nondominated, Pareto optimal, if there is no other member of the domain which does as well as *x'* and better at least in one of the objectives. This means that, *x'* is said to be a Pareto solution iff there doesn't exist another *x** in the feasible region so that $f_i(x^*) \le f_i(x')$ for all *i* and strictly for at least one *i*.

The image of the Pareto set under the objective function in the objective space is known as Pareto front.

B. *Evolutionary Algorithm*

Evolutionary algorithms are population based solution methods which work by attempting to 'evolve' a good solution to the problem at hand. A 'population' of candidate solutions is kept while new solutions are generated in the neighborhood of the existing population, and poor solutions are removed from the population. By favouring better solutions, either by letting them live longer, or giving them more chances to create 'child' solutions, the population can be made to move towards regions containing better solutions. The algorithm involves the following steps

1. Generate random initial population
2. Choose some members using the fitness value and use a crossover and mutation operator on the chosen members to generate a child population. Then construct a new population of the same number of as the original by choosing the fittest from the parent set and the children.
3. If termination criteria is met stop, else using the new set as a parent population go back to step 2.

The commonly used evolutionary algorithm for multiobjective optimization problems is genetic algorithm in which each population is expressed as a chromosome using 0's and 1's. [10].

C. *Fuzzy Preference*

Fuzzy logic is a logic based on the idea that all things admit of degrees. It is the extension of the Boolean logic. In Boolean logic, an element is either a member of a given set or not. But in fuzzy logic, an element can have some degrees of membership for a set, which lies between being a member and not. [8]

Let $f_A(x)$ be a membership function of x to set A.

In Boolean logic:

$$f_A: X \to \{1,0\},\ \text{by}\ f(x)=\begin{cases}1, & \text{if } x \in A\\ 0, & \text{if } x \notin A\end{cases}$$

where A is a set and X is the universal set.

In fuzzy logic:

$f_A: X \to [0,1]$, $f_A(x)$ is the degree of x in set A,

where A is a set in the universal set X.

$f_A(x) = 1$, means x is totally in A while if $f_A(x)=0$ means x is not in A. Furthermore $f_A(x)$ can have any value between 0 and 1, known as the degree of membership of x in set *A*.

Unlike crisp or Boolean set, fuzzy set doesn't put a solid boundary between sets. Rather it reflects how people think. For instance consider a set, say A, with elements of tall men. In crisp set theory there will be a solid boundary on the membership of the set. For example, a man more than or equal to 190cm tall is the member of set A, with membership function 1 and a man with height less than 190 cm is not a member and has membership function 0, as shown in Figure 1(a).

But in the fuzzy set theory the membership function for those members 190cm tall and more will be 1; and it keeps on decreasing as the height becomes smaller than 190cm and finally become 0 after some point onwards, as shown in Figure 1(b).

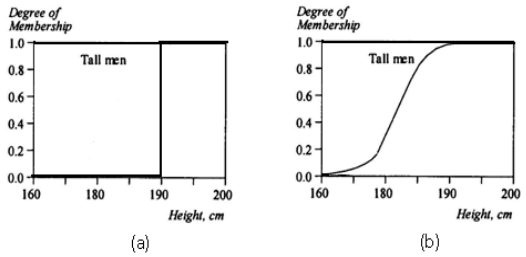

Fig. 1 The membership function for set of tall men in (a) crisp set (b) fuzzy set

A conditional tradeoff of objective function i, $f_i(x) = y_i$, for a unit decrease of objective function j, $f_j(x) = y_j$, is the amount of objective i which the decision maker is willing to give up for a unit decrease on objective j while all other objectives remains the same. If conditional tradeoff of objective 1 for a unit decrease of objective 2 is b, this means, $(y_1, y_2, y_3, \ldots, y_n)$ is equivalent to $(y_1+b, y_2-1, y_3, \ldots, y_n)$

Assigning such a tradeoff needs subjective judgment and depends on the decision maker's preference and it is not an easy task. Rather than assigning an exact tradeoff it is easier to give the preference in fuzzy way. For a unit decrease of objective 2 the decision maker is willing to give around b units of objective 1, which means

$(y_1, y_2, y_3, \ldots, y_n)$ is equivalent to $(y_1+k, y_2-1, y_3, \ldots, y_n)$, for $k \le b$.

As k gets larger the acceptability or the equivalency keep on decreasing and become zero after some limit onwards, say after $k=b + d$ as shown in Figure 2. Let's define d to be the width and b the average tradeoff.

It is possible to consider the acceptability as a membership function as of in the fuzzy set theory. So acceptable means membership value 1 and unacceptable means membership value 0 and will have values between 0 and 1 depending of degree of acceptability.

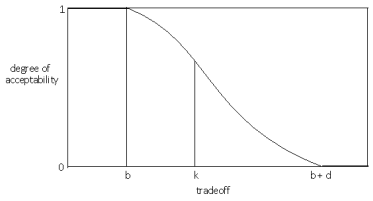

Fig. 2 Fuzzy tradeoff

III. INCORPORATING PREFERENCE WITH EVOLUTIONARY ALGORITHM

To solve a multiobjective optimization problem and to give out solutions for the decision maker which incorporates the decision maker's preference first it is necessary to get the decision maker's preference. For each couple of objectives, we ask the decision maker to give the fuzzy tradeoff of one objective over the other.

Suppose that for objective functions $f_i(x)$ and $f_j(x)$ with different i and j, the decision maker give us the average tradeoff (a_{ij}) and the width (b_{ij}).

Hence we have two matrices $A= (a_{ij})$ and $B= (b_{ij})$. It is meaningless of computing the tradeoff of the i^{th} function for a unit decrease of the i^{th} function itself. Hence a_{ii} is a junk number, so let's put $a_{ii} = 0$ and $b_{ii} = 0$, so that its impact in constructing the weight range will be omitted.

From A, it is possible to calculate the average weight, a weight with high degree of acceptability as follows.

$$\overline{w}_p = \frac{\sum_{\substack{i=1 \\ i\neq p}}^{k} a_{pi}}{\sum_{j=1}^{k}\sum_{\substack{i=1 \\ i\neq j}}^{k} a_{ij}} = \frac{\sum_{i=1}^{k} a_{pi}}{\sum_{j=1}^{k}\sum_{i=1}^{k} a_{ij}}$$

$$\overline{w} = \begin{pmatrix} \overline{w}_1 \\ \overline{w}_2 \\ . \\ . \\ . \\ \overline{w}_k \end{pmatrix}$$ is the average weight for the k objective functions.

It is also possible to take the average normalized fuzzy width as follows:

$$\overline{b}_p = \frac{\left(\sum_{\substack{i=1 \\ i\neq p}}^{k} b_{pi} \Big/ (k-1) \right)}{\sum_{i=1}^{k} \left(\sum_{\substack{j=1 \\ j\neq i}}^{k} (b_{ij}) \Big/ (k-1) \right)} = \frac{\sum_{j=1}^{k} b_{pj}}{\sum_{i=1}^{k}\sum_{j=1}^{k} b_{ij}}$$

$$\overline{b} = \begin{pmatrix} \overline{b}_1 \\ \overline{b}_2 \\ . \\ . \\ . \\ \overline{b}_k \end{pmatrix}$$ is the normalized average fuzzy width.

For each function $f_i(x)$, the fuzzy weight, w_i, is around the corresponding average weight with some degree of acceptability, as shown Figure3.

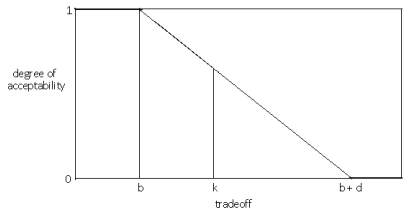

Fig. 3 Fuzzy weight

After getting the weight boundaries for each objective function the next step is to generate a random weight from the given fuzzy preference in such a way that a

number with high acceptability needs to have high probability, as shown in Figure 3.

The acceptability function can have different path by joining the points (b, 1) and ($b+d$, 0). The path doesn't have a significant impact. Perhaps it is easier to consider a straight line joining the two points, say $g_i(x)$, for the i^{th} function. Then it is necessary to generate random weight under the line. For such purpose it is possible to use different methods like the inversion method of sampling. And the condition is that the area under the line should be 1, to make it a probability density function. In order to make the area 1, it may be necessary to adjust the line. Suppose the line passes thorough (b, y) and ($b+d$, 0) for some y, not necessarily 1.

$$\text{Area} = \frac{1}{2} \times y \times d = 1$$

$$y = \frac{2}{d}$$

$$\text{Hence } g_i(x) = \frac{-2x}{d^2} + \frac{2(b+d)}{d^2}, \text{ for all } i.$$

In other words, w_i is a random variable with probability density function $g_i(x)$.

In the evolutionary algorithm we can incorporate this fuzzy preference and zoom to the area in the Pareto front to which the decision maker is interested. To do that we construct the fitness function in each iteration, by taking the weighted sum of the objective functions.

$$\text{The fitness function} = \sum_{i=1}^{k} w_i(j) f_{i(x)}$$

where $w_i(j)$ is the weight of f_i at iteration j.

The conditional fuzzy tradeoff depends on the current value of the objective functions. For two different points the decision maker may give different fuzzy tradeoffs. So it is better to get back to the decision maker after a number of iterations of the algorithm and update the fuzzy weight. Hence the fuzzy weight may vary on the process till the decision maker is satisfied with the solution or no further achievements can be made.

The Proposed Algorithm

(1) Set j = 0, number of generation
Randomly generate initial population, $P(0)$.

(2) Generate $w_i(j)$, with a probability density function $g_i(x)$, for all $i \in \{1,2,...,k\}$, k is number of objective functions.

(3) Select a set of parents, $Par(j)$ from $P(j)$

(4) Generate new members using crosover and mutation, $C(j)$

(5) Creat a new population, $P(j+1)$,
by replacing some memebers of $P(j)$ by $C(j)$
set $j = j + 1$

(6) If the termination criteria fulfills stop
else go back to stape (2).

IV. Experiment and Results

We simulate the proposed algorithm using MATLAB. For the simulation purpose we use the bi-objective function $F(x)=(f_1(x), f_2(x))$, which are used as a test function in previous studies.

$$\min F(x) = (f_1(x), f_2(x))$$

$$\text{s.t.} \quad x \in [0,1]$$

The test functions used are ([8], [9]):

1. $f_1(x) = \frac{1}{n}\sum_{i=1}^{n} x_i^2$ and $f_2(x) = \frac{1}{n}\sum_{i=1}^{n} (x_i - 2.0)^2$

2. $f_1(x) = x_1$ and $f_2(x) = g(x) \times (1.0 - \sqrt{\frac{f_1(x)}{g(x)}})$,

$$\text{where } g(x) = g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

3. $f_1(x) = x_1$ and $f_2(x) = g(x) \times (1.0 - (\frac{f_1(x)}{g(x)})^2)$,

$$\text{where } g(x) = g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

4. $f_1(x) = x_1$ and $f_2(x) = g(x) \times (1.0 - \sqrt[4]{\frac{f_1(x)}{g(x)}} - (\frac{f_1(x)}{g(x)})^4)$,

$$\text{where } g(x) = g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

5. $f_1(x) = x_1$ and $f_2(x) = g(x) \times (1.0 - \sqrt{\frac{f_1(x)}{g(x)}} - (\frac{f_1(x)}{g(x)})\sin(10\pi f_1(x)))$,

$$\text{where } g(x) = g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

The first two test functions have a convex Pareto front, the third one a concave Pareto front, the fourth a concave and convex Pareto front and the last one with discontinuous Pareto front.

For all test function n=2, so that we can compare the results with previous works ([8], [9]). Furthermore we take the probability of mutation to be 0.1 and the probability of reproduction 0.9; and the preference of the decision maker, the fuzzy average weight and the width, as been taken as; 1.4, 0.25 for one of the functions and 0.2, 0.15 for the other function, respectively. By switching the fuzzy average weight and the width we have run the program in favor of both functions. We run the program 50 times and record the results as follow.

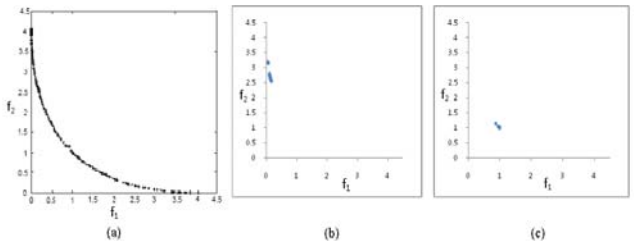

Fig. 4 (a) The result from the previous methods for the first test function ([8],[9]); (b) Result on the first test function with high fuzzy weight for function 1 (c) Result on the first test function with high fuzzy weight

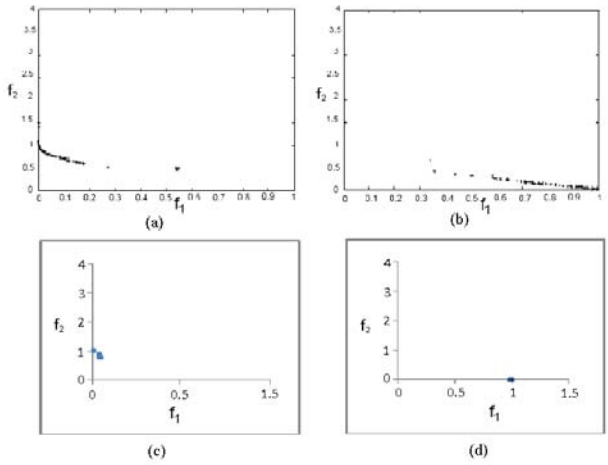

Fig. 5 (a) and (b) The result from the previous methods for the second test function ([8], [9]); (c) Result on the second test function with high fuzzy weight for function 1 (d) Result on the second test function with high function with high fuzzy weight for function 2.

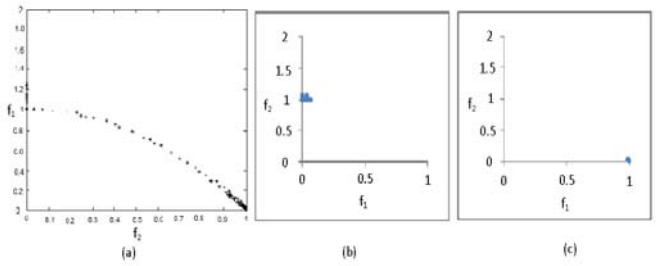

Fig. 6 (a) The result from the previous methods for the third test function ([8], [9]); (b) Result on the third test function with high fuzzy weight for function 1 (c) Result on the third test function with high fuzzy weight for function 2.

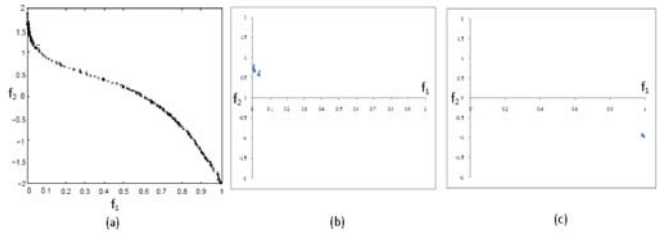

Fig. 7 (a) The result from the previous methods for the fourth test function [9]; (b) Result on the fourth test function with high fuzzy weight for function 1 (c) Result on the fourth test function with high fuzzy weight for function 2.

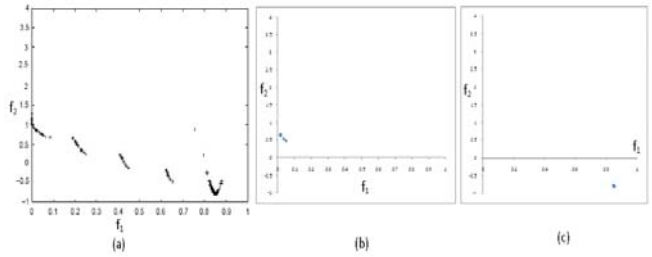

Fig. 8 (a) The result from the previous methods for the fifth test function [9]; (b) Result on the fifth test function with high fuzzy weight for function 1 (c) Result on the fifth test function with high fuzzy weight for function 2.

As shown above from the simulation results, by incorporating the fuzzy preference of the decision maker, it is possible to zoom out further and get solutions according to the decision maker's preference. In the first run, a high weight given to the first objective function, f_1, and as shown the points converge to the left side where the smaller value of the first objective function is found and when high weight is given to the 2nd objective function the points converges downward.

V. Conclusion

In this study, the incorporation of fuzzy preference of the decision maker has been discussed and simulation on test functions has been done. From the simulation results it is clear to see that by incorporating the fuzzy tradeoffs of the decision maker's preference we can generate solutions in which the decision maker is interested. In generating the fuzzy weight we use the fuzzy tradeoffs given by the decision maker. The preference of the decision maker as a fuzzy tradeoffs may vary with the functional values. So, to generate the weight interval it might be necessary to generate the weight interval interactively while the program is running. This paper shows how to generate the weight interval from the fuzzy tradeoffs and how to embed it in the evolutionary algorithm.

Acknowledgment

This work was supported in part by the U.S.M short terms grant no. 304/PMATH/639036. The first author would like to acknowledge a support from TWAS-USM fellowship, 2008.

References

[1] Ehrgott, M.: *Multicriteria Optimization*, Springer/Berlin, 2005.

[2] Chankong, V., & Haimes, Y.: *Multiobjective Deci-sion Making: Theory and Methodology*, North-Holland/New York, 1983.

[3] Mavrotas, G.: Effective implementation of the ε-constraint me-thod in Multi-Objective Mathematical Programming problems, *Applied Mathematics and Computation;* 213, 2, pp. 455-465, 2009.

[4] Keeney, R. L., & Raiffn, H.: *Decision with Multiple Objectives: Preferences and Value Tradeoffs*, New York/ John Wiley & Sons, Inc. 1976.

[5] Emelichev, V. A., Kravtsov, M. K., & Yanushkevich, O. A.: *Lexicographic optima in the multicriteria discrete optimization problem*, Mathematical Notes; 58, 3, pp. 928-932, 1995.

[6] Zitzler, E., Laumanns, M., & Bleuler, A. S.: *A Tutorial on Evolutionary Multiobjective Optimization*, Switzerland: Swiss Federal Institute of Tech-nology (ETH) Zurich, Computer Engineering and Networks Laboratory (TIK),.

[7] Coello, C. A.: *Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored*, Higher Education Press, co-published with Springer-Verlag GmbH 2009, 18-30.

[8] Jin, Y., & Sendhoff, B., Incorporating of Fuzzy Preferences into Evolutionary Multiobjective Optimization, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002, 1, 26-30, Singapore.

[9] Yaochu, J., Markus, O., & Bernhard, S.: Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? *Proc. Genetic and evolutionary computation*, 2001, 1042-1049. San Francisco: CA.

[10] Negnevitsky, M.: *Artificial Intelligence: A Guide to Intelligent Sysytem,*. Henry Ling Limited/Harlow, England, 2005.