INTEGRATING FUZZY PREFERENCE IN GENETIC ALGORITHM TO SOLVE MULTIOBJECTIVE OPTIMIZATION PROBLEMS

ONG HONG CHOON and SURAFEL LULESEGED TILAHUN

School of Mathematical Science
Universiti Sains Malaysia
11800 USM, Penang
Malaysia
e-mail: hcong@cs.usm.my
surafelau@yahoo.com

Abstract

Population based solution methods have become more common in solving multiobjective optimization problems. These methods overcome the limitation of classical methods of finding a single solution with a single run of the algorithms. The studies of incorporating the decision maker’s preference to genetic algorithm had done so far use binary importance comparison of objectives, which lacks uniformity. This study shows how to incorporate the fuzzy preference of a decision maker in evolutionary algorithm so that the solutions will zoom to the region in which the preference of the decision maker lies. Cumulative fuzzy tradeoffs are expressed using an appropriate probability distribution function which agrees with the fuzzy membership function. It will then be incorporated in the fitness evaluation stage of genetic algorithm. From the simulation result on selected test functions and comparing this with previous studies, it is shown that incorporating the decision maker’s fuzzy preference will give solutions which satisfy the decision maker’s preference better.

2010 Mathematics Subject Classification: 90B50.

Keywords and phrases: multiobjective optimization, fuzzy preference, genetic algorithm.

Received March 14, 2011
I. Introduction

Multiobjective optimization problems are optimization problems with more than one and usually conflicting objective functions. Finding a solution from possible set of actions is very difficult due to the conflict of objectives. Optimizing one of the objective functions after a certain level results in the worsening of other competing objectives. The older and almost direct forward approach in solving these problems is to change the problems into single objective optimization problems. One among the most commonly used method 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 a real valued function to optimize will be constructed. Trade-off method ([2-3]) is also a method of changing the multiobjective optimization problem into single objective optimization problem, where all the objective functions except one will be put into a constraint set by putting maximum affordable values for minimization problem. There are many 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. These solution methods will give a single Pareto solution. Sometimes it is better to give a set of possible solutions according to the decision maker’s subjective preference.

Population based metaheuristic solution methods have the advantage that it gives multiple solutions within a single run of the algorithm. Even though it doesn’t guarantee Pareto optimality it will give reasonable solutions. Genetic algorithm is an evolutionary algorithm which mainly use solution algorithm in solving optimization problems. It is inspired by natural evolution proposed by Darwin [6]. In the algorithm an initial population of solutions is 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 algorithms 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 order of importance 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 genetic algorithm. In Section II, basic concepts and preliminaries will be discussed followed by Section III which shows how the cumulative fuzzy preference is calculated and appropriate probability distribution constructed. Furthermore, in Section III, the embedding of the fuzzy preference in the fitness evaluation stage of the algorithm will be explained. Simulation results and comparison of the results with previous studies will be done in Section IV. Finally, in Section V, we give a conclusion.

II. Problem Statement and Preliminaries

A. Multiobjective optimization

Multiobjective optimization problems can be found in various fields, wherever optimal decisions are needed to be taken. 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 \( F : \mathbb{R}^n \rightarrow \mathbb{R}^k \), \( F(x) = (f_1(x), f_2(x), ..., f_k(x)) \) and \( S \subseteq \mathbb{R}^n \).

Then

\[
\min_{x \in S \subseteq \mathbb{R}^n} F(x) = (f_1(x), f_2(x), ..., f_k(x))
\]

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 if and only if there does not exist another \( x^* \) in the feasible region so that \( f_i(x^*) \leq f_i(x') \) for all \( i \) and strictly for at least one \( i \).
Example 1. Consider \( F(x) = (f_1(x), f_2(x)) \) and \( S = \{x_1, x_2, x_3\} \) and suppose the image of \( S \) under the objective function is \( F(x_1) = (f_1(x_1), f_2(x_1)) = (0, 2) \), \( F(x_2) = (f_1(x_2), f_2(x_2)) = (1, 1) \) and \( F(x_3) = (f_1(x_3), f_2(x_3)) = (1, 3) \).

In the minimization problem which we are considering, \( x_1 \) is doing better for \( f_1(x) \) and \( x_2 \) is the best for \( f_2(x) \). And clearly \( x_1 \) and \( x_2 \) do better for both functions than \( x_3 \). But we cannot compare \( x_1 \) and \( x_2 \), because \( x_1 \) is better in terms of the first function and \( x_2 \) is better in terms of the second. Hence \( x_1 \) and \( x_2 \) are called Pareto optimal solutions.

The image of the Pareto set 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 [10]. 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. It does not guarantee that the final solutions will be Pareto optimal. However it will give a sound and acceptable 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 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 [11].

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:

\[
\begin{align*}
& f_A : X \to \{1, 0\}, \\
& x \in A & f(x) = 1, \\
& x \notin A & f(x) = 0,
\end{align*}
\]

where \( A \) is a set and \( X \) is the universal set.

In fuzzy logic:

\[
\begin{align*}
& f_A : X \to [0, 1], \\
& f_A(x) \text{ is the degree of } x \text{ in set } A,
\end{align*}
\]

where \( A \) is a set in the universal set \( X \).

The functional value \( f_A(x) = 1 \), means \( x \) is totally in \( A \) while \( f_A(x) = 0 \) 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 \( A \).

\[\text{Figure 1. The membership function for set of tall men in (a) crisp set (b) fuzzy set.}\]

Unlike crisp or Boolean set, fuzzy set does not 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).
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 remain the same. If conditional tradeoff of objective $i$ for a unit decrease of objective $j$ is $b$, this means,

$$ (y_1, y_2, ..., y_i, ..., y_j, ..., y_k) \sim (y_1, y_2, ..., y_i + b, ..., y_j - 1, ..., y_k). $$

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, as for a unit decrease of objective $j$ the decision maker is willing to give around $b$ units of objective $i$, which means

$$ (y_1, y_2, ..., y_i, ..., y_j, ..., y_k) \sim (y_1, y_2, ..., y_i + w, ..., y_j - 1, ..., y_k), $$

for $w \leq b + d$ for some $d$, with some membership function as shown in Figure 2.

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

![Figure 2. Fuzzy tradeoff.](image)

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 on the degree of acceptability.

### III. Incorporating Fuzzy 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 to
compute 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 us 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:

\[
\bar{w}_p = \frac{\sum_{i=1}^{k} a_{pi}}{\sum_{i=1}^{k} \sum_{j=1}^{i\neq j} a_{ij}} = \frac{\sum_{i=1}^{k} a_{pi}}{\sum_{i=1}^{k} \sum_{j=1}^{i\neq j} a_{ij}},
\]

where \( \bar{w} = \left( \begin{array}{c} \bar{w}_1 \\ \bar{w}_2 \\ \vdots \\ \bar{w}_k \end{array} \right) \) is the average weight for the \( k \) objective functions.

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

\[
\bar{b}_p = \frac{\sum_{i=1}^{k} b_{pi} / (k - 1)}{\sum_{i=1}^{k} \sum_{j=1}^{i\neq j} (b_{ij}) / (k - 1)} = \frac{\sum_{i=1}^{k} b_{pi}}{\sum_{i=1}^{k} \sum_{j=1}^{i\neq j} b_{ij}},
\]

where \( \bar{b} = \left( \begin{array}{c} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_k \end{array} \right) \) 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 Figure 3.

![Figure 3. Fuzzy weight for \( i \)th function.](image)

After obtaining 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 \((b_i, 1)\) and \((b_i + d_i, 0)\). The path does not have a significant impact. Perhaps it is easier to consider a straight line joining the two points, say \( g_i(w) \), 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. 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 \((b_i, y)\) and \((b_i + d_i, 0)\) for some \( y \), not necessarily 1.

\[
\text{Area} = \frac{1}{2} \times y \times d_i = 1
\]

\[
y = \frac{2}{d_i}.
\]

Hence \( g_i(w) = -\frac{2w}{d_i^2} + \frac{2(b_i + d_i)}{d_i^2} \) for all \( i \) and \( d_i \) is the average width.

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

It is also possible to use the right part of normal probability distribution, as \( w_i \sim N(\mu_i, \sigma_i^2) \) and \( w_i \geq \mu_i \), where \( \mu_i \) is \( b_i \) and \( \sigma_i \) is equivalent to \( \frac{d_i}{3} \).
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.

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.

---

**Fuzzy Preference Incorporated Genetic Algorithm**

**Input**

\[ f_j(x), \ x \in \mathbb{R}^n, \ j \in \{1, 2, ..., m\} \leftarrow \text{Objective functions} \]

\[ P_r, P_m \ (\text{probability of reproduction and mutation}) \leftarrow \text{Algorithm Parameters} \]

\[ g_j(w) \leftarrow \text{Probability density function for the dynamic weight} \]

**Output**

\[ x_j, \ i \in \{1, 2, ..., k\} \leftarrow \text{solutions for the min problem} \]

**Begin**

for \( i = 1 : k \)

\[ x_i \leftarrow \text{Generate initial population.} \]

end for

**Repeat**

for \( j = 1 : m \)

\[ w_j \leftarrow \text{Generate weight from } g_j(w) \]

end for
\begin{align*}
\text{Calculate fitness} \\
\end{align*}

\begin{align*}
&\text{for } i = 1 : k \\
&\quad \text{fun}_i = \sum_{j=1}^{m} w_j f_j(x_i) \quad \text{Calculate fitness} \\
&\end{align*}

\begin{align*}
&\text{Choose the fittest} \\
&\quad \text{for } i = 1 : k' \quad k' \text{ is round } (k/4) \\
&\text{if } (\text{rand} \leq P_r) \\
&\quad x'_i, x''_i \quad \text{Select parents from the fittest.} \\
&\quad y_i, y_j \quad \text{reproduction.} \\
&\text{else if} \\
&\quad y_i = x'_i; y_j = x''_i \\
&\text{end if} \\
&z = (x', y) \quad \text{Put parents and children together} \\
&\text{for } i = 1 : k \\
&\quad \text{if } (\text{rand} \leq P_m) \\
&\quad z_i \quad \text{mutation} \\
&\text{end if} \\
&\text{end for} \\
&z = \text{sort}(\text{fun}(x, z)) \\
&\text{for } i = 1 : k \\
&\quad x_i = z_i^{-1} \quad \text{inverse } z \text{ of fun (preimage of fun)} \\
&\text{end for} \\
&\text{Until stopping condition meet} \\
&\text{end}
IV. Experimental 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.

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

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

The test functions used are taken from ([8-9]). 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 15 times.

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

Since $n = 2$ our functions are:

$$f_1(x) = \frac{x_1^2 + x_2^2}{2} \text{ and } f_2(x) = \frac{(x_1 - 2.0)^2 + (x_2 - 2.0)^2}{2}.$$ 

The Pareto front is a convex, for $n = 2$.

![Figure 4](image1.png)

**Figure 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 for function 2.
2. \( f_1(x) = x_1 \) and \( f_2(x) = g(x) \left( 1.0 - \frac{f_1(x)}{g(x)} \right) \),

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

Here we have a convex Pareto front. For \( n = 2 \):

\[
\begin{align*}
    f_1(x) &= x_1 \\
    f_2(x) &= (1 + 9x_2) \left( 1.0 - \frac{x_1}{\sqrt{1 + 9x_2}} \right).
\end{align*}
\]

Figure 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.

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

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

Here we have a concave Pareto front determined by the following two functions, for \( n = 2 \):

\[
\begin{align*}
    f_1(x) &= x_1 \\
    f_2(x) &= (1 + 9x_2) \left( 1.0 - \left( \frac{x_1}{1 + 9x_2} \right)^2 \right).
\end{align*}
\]
4. $f_1(x) = x_1$ and $f_2(x) = g(x) \left( 1.0 - 4 \sqrt{\frac{f_1(x)}{g(x)}} - \left( \frac{f_1(x)}{g(x)} \right)^4 \right)$,

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

The Pareto front of these functions will be partly convex and partly concave. The functions for $n = 2$ is given by:

$$f_1(x) = x_1 \quad \text{and} \quad f_2(x) = (1 + 9x_2) \left( 1.0 - 4 \sqrt{\frac{x_1}{1 + 9x_2}} - \left( \frac{x_1}{1 + 9x_2} \right)^4 \right).$$

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

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

**Figure 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 the second function.

**Figure 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.
Unlike the above function here we have a discontinuous Pareto front, with functions given as:

\[ f_1(x) = x_1 \quad \text{and} \quad f_2(x) = (1 + 9x_2)^2 \left( 1 - \frac{x_1}{1 + 9x_2} - \frac{x_1}{1 + 9x_2} \sin(10\pi x_1) \right). \]

![Figure 8](image)

**Figure 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, \( f_1 \), 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 converge downward.

### V. Conclusion

This paper discusses the computation of cumulative fuzzy preference from conditional fuzzy tradeoffs. Furthermore, using the fuzzy preference as a dynamic weight for objective function and how to embed on the fitness evaluation stage is also explained and implemented. According to the simulation results and comparison with previous studies it is clear that by incorporating this fuzzy preference it is possible to get solutions which satisfy the decision maker better. The preference of the decision maker as fuzzy tradeoffs may vary with the functional values. So, to generate the weight interval it might be necessary to update the preference interactively while the program is running. This study shows how to generate a dynamic weight from the fuzzy tradeoffs using user defined probability distribution or possibly normal distribution and how to embed it in the evolutionary algorithm.
Acknowledgement

The second author would like to acknowledge a support from TWAS-USM fellowship, 2008.

References