CHAPTER 7

Satisficing solutions of fuzzy multiobjective optimization problem
CHAPTER 7
Satisficing solutions of fuzzy multiobjective optimization problem

7.1 Background and motivation

Optimization is the task of finding one or more solutions which correspond to minimizing (or maximizing) one or more specified objectives and which satisfy all constraints (if any). A single-objective optimization problem involves a single objective function and usually results in a single solution, called an optimal solution. On the other hand, a multiobjective optimization task considers several conflicting objectives simultaneously.

Multiobjective optimization is a rapidly growing area of research and application in modern-day optimization. Multiobjective optimization (also called multicriteria optimization, multiperformance, or vector optimization) can be defined as the problem of finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term optimize means finding such a solution which would give the values of all the objective functions acceptable to the designer [101].

The fuzziness of this area lies in the fact that there is no accepted definition of optimum as in single-objective optimization. Hence it is difficult to even compare the results of one method to another's because, normally, the decision about the best answer corresponds to the so-called (human) decision maker. Thus, one is forced to look for possible compromises and finally decide which one to implement.
Multiobjective optimization has been available for about two decades, and its application in real-world problems is continuously increasing. In contrast to the excess of techniques available for single-objective optimization, relatively few techniques have been developed for multiobjective optimization. In single-objective optimization, the search space is often well defined. As soon as there are several possibly contradicting objectives to be optimized simultaneously, there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality. When we try to optimize several objectives at the same time the search space also becomes partially ordered. To obtain the optimal solution, there will be a set of optimal trade-offs between the conflicting objectives.

The meaning of an optimum has to be redefined for multiobjective optimization problem. The definition of optimality in multiobjective optimization is not simple. Due to presence of conflicting objectives, improvement in one objective may cause deterioration in another objective. For example, maximization of the structural stability of a mechanical structure may cause an increase in costs, working against the additional objective to minimize costs. Tradeoffs exist between such conflicting objectives, and the task is to find solutions that balance these trade-offs. Such a balance is achieved when a solution cannot improve any objective without degrading one or more of the other objectives. It is very clear that if there are two objectives to be optimized, it might be possible to find a solution which is the best with respect to the first objective, and another solution, which is the best with respect to the second objective.

Even though some real-world problems can be reduced to a matter of a single objective, very often it is hard to define all the aspects in terms of a single objective. Defining multiple objectives often gives a better idea of the task. Multiobjective optimization is the art of detecting and making good compromises. It bases upon the fact that most real-world decisions are compromises between partially conflicting objectives that cannot easily be offset against each other. It is without a doubt a very important research topic both for scientists and engineers, not only because of the multiobjective nature of most real-world problems but also because there are still many open questions in this area. In
operations research, more than 20 techniques have been developed over the years to try to
deal with functions that have multiple objectives, and many approaches have been
suggested, going all the way from naively combining objectives into one, to the use of
game theory to coordinate the relative importance of each objective.

There are two general approaches to multiple-objective optimization. One is to combine
the individual objective functions into a single composite function or move all but one
objective to the constraint set. In the former case, determination of a single objective is
possible with methods such as utility theory, weighted sum method, etc., but the problem
lies in the proper selection of the weights or utility functions to characterize the decision-
maker's preferences. In practice, it can be very difficult to precisely and accurately select
these weights, even for someone familiar with the problem domain. Compounding this
drawback is that scaling amongst objectives is needed and small perturbations in the
weights can sometimes lead to quite different solutions. In the latter case, the problem is
that to move objectives to the constraint set, a constraining value must be established for
each of these former objectives. This can be rather arbitrary. In both cases, an
optimization method would return a single solution rather than a set of solutions that can
be examined for trade-offs. For this reason, decision-makers often prefer a set of good
solutions considering the multiple objectives. The second general approach is to
determine an entire Pareto optimal solution set or a representative subset.

In single objective optimization, one attempts to obtain the best design or decision, which
is usually the global minimum or the global maximum depending on the optimization
problem is that of minimization or maximization. In case of multiple objectives, there
may not exist one solution which is best (global minimum or maximum) with respect to
all objectives. In a typical multiobjective optimization problem, there exists a set of
solutions which are superior to the rest of solutions in the search space when all
objectives are considered but are inferior to other solutions in the space in one or more
objectives. These solutions are known as Pareto optimal solutions or nondominated
solutions. The corresponding objective vectors in objective space are referred to as the
Pareto front. The rest of the solutions are known as dominated solutions.
Since none of the solutions in the nondominated set is absolutely better than any other, any one of them is an acceptable solution. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s). Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems since the final solution of the decision-maker is always a trade-off. Pareto optimal sets can be of varied sizes, but the size of the Pareto set usually increases with the increase in the number of objectives.

The first notion of optimality in this setting goes back to Edgeworth [26] in 1881 and Pareto [108] in 1896 and is still the most widely used. In Pareto optimality every feasible alternative that is not dominated by any other in terms of the component wise partial order is considered to be optimal. Hence each solution is considered optimal that is not definitely worse than another. Thus, multiobjective optimization does not yield a single or a set of equally good answers, but rather suggests a range of potentially very different answers.

7.2 Multiobjective genetic algorithms

Genetic algorithms have been successfully applied to many optimization problems for which it is difficult to find exact optimal solutions by conventional optimization methods. The reasons for the failure of using a conventional exact method include, most formally, the problem is too complex or too large in size, the objective functions are highly non-linear or non-differentiable, the problem has many local optima, and the solution space is non-convex or disconnected. Actually, there are other heuristic methods, such as simulated annealing, or tabu search, that can facilitate the solutions of these complex problems. However, these methods are more suitable for single objective optimization problems since they deal with just one solution at a time. The notion of Pareto optimality is only one consideration in determining the optimal solutions of a multiobjective optimization problem.
In trying to solve multiobjective optimization problems, many traditional methods scalarize the objective vector into a single objective. In those cases, the obtained solution is highly sensitive to the weight vector used in the scalarization process and demands the user to have knowledge about the underlying problem. Moreover, in solving multiobjective problems, decision maker may be interested in a set of Pareto-optimal points, instead of a single point. Since genetic algorithms work with a population of points, it seems natural to use genetic algorithms in multiobjective optimization problems to capture a number of solutions simultaneously.

Being a population-based approach, genetic algorithms are well suited to solve multiobjective optimization problems. Generic single-objective genetic algorithms can be modified to find a set of multiple non-dominated solutions in a single run. The ability of genetic algorithms to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. The crossover operator of genetic algorithms may exploit structures of good solutions with respect to different objectives to create new nondominated solutions in unexplored parts of the Pareto front. In addition, most multiobjective genetic algorithms do not require the user to prioritize, scale, or weight objectives. Therefore, genetic algorithms have been the most popular heuristic approach to multiobjective design and optimization problems. Jones et al. [57] reported that 90% of the approaches to multiobjective optimization aimed to approximate the true Pareto front for the underlying problem. A majority of these used a meta-heuristic technique, and 70% of all metaheuristics approaches were based on evolutionary approaches. Some best known approaches using genetic algorithms to deal with the multiobjective optimization problems includes plain aggregating approach, the population based non Pareto approach, the Pareto based approach, and the niche induction approach and are discussed below.

**Plain aggregating approach:** Since a genetic algorithm usually requires a scalar of each individual to determine its fitness, the objectives need to be combined somehow into a single fitness. One way is to aggregate multiple objectives into one value using a weight
vector such that a multiobjective optimization problem is reduced to a single objective optimization problem. One problem with this approach is that objectives are problem dependent and thus require the specific knowledge before a final decision can be made. If we simply aggregate the objective values using some weights, then certain aspects of the problem may be left out of consideration. An inappropriate setting of the weight vector may lead to sub-optimality.

**Population based non Pareto approach:** The approach has been studied by Schaffer [137], Fourman [30], and Kursawe [68]. In this approach, the selection/reproduction is performed by treating the non-commensurable objective functions separately. Schaffer [137] was probably the first who studied multiobjective optimization problems using this approach with genetic algorithms. Population based non Pareto approach in general attempts to generate multiple non-dominated solutions. However, it does not make direct use of the actual definition of Pareto optimality. This approach tends to generate solutions with one extremely good objective.

**Pareto based approach:** Pareto based fitness assignment was first proposed by Goldberg [35]. He suggested that individuals that are non-dominated should be given equal chance for reproduction. The non-dominated solutions are assigned rank 1 and temporarily removed from consideration. Then, a new set of non-dominated individuals are determined and ranked 2, and so on. Fonseca and Fleming [29] proposed a different scheme in which an individual’s rank corresponds to the number of individuals in the current population by which it is dominated. Tournament selection based on dominance properties has been proposed by Horn et al. [54]. One advantage of Pareto ranking is that it is not sensitive to the convexity or non-convexity of the trade-offs surface which has some impact on the solutions. Another advantage is that, because it rewards good performance in each objective dimension regardless of others, solutions that exhibit good performance in many, if not all, objective dimensions are most likely to be produced by crossover and mutation operations.
Niche induction approach: Although Pareto-based ranking correctly assigns all non-dominated solutions the same fitness, it does not guarantee that the Pareto optimal set be uniformly sampled. Therefore, the population will tend to converge to only one of Pareto optima due to stochastic errors in the selection process. This phenomenon is called genetic drift [35]. Fitness sharing is, therefore proposed to help alleviate this problem as suggested by Goldberg [35], Fonseca and Fleming [29], and Srinivas and Deb [146]. The idea of fitness sharing is that the fitness of individuals that are in a very crowded neighbourhood should be reduced such that these individuals get lower chance of reproduction whereas the individuals in a less crowded neighbourhood should receive higher chance of reproduction. The sharing can be performed either on the decision variable domain or on the objective domain.

7.3 Multiobjective optimization problem with fuzzy relation equations

Fuzzy relation is a generalization of the Boolean relation [64]. For a system defined by fuzzy relation equations, the input and output parameters of the system are somehow related. A decision maker is expected to set the input parameters to certain approximate levels such that a desired output is obtained while several objective functions are simultaneously optimized. Such problem can be formulated as a multiobjective optimization problem with fuzzy relation equation constraints [153].

Wang [153] first studied the problem of multiobjective mathematical programming for medical applications with constraints defined by max-min composite fuzzy relation equations. Feasible region being normally non-convex, the properties of the efficient points of a non-convex feasible region under multiobjectives were investigated. A procedure was proposed that transforms these efficient points of an interval-valued decision space into a constant-valued decision space when the level of confidence is given by a decision maker. Then the transformed problem becomes a multi-attribute decision problem and the optimal alternative is found. Unfortunately, this work required the objectives of the multiobjective optimization problem to be linear. Moreover, it
required the knowledge of all minimal solutions of system of fuzzy relation equations, which is not trivial at all.

Loetamonphong et al. [78] studied class of optimization problems with multiple objective functions subject to a set of max-min fuzzy relation equations. Since the feasible domain of such a problem is in general non-convex and the objective functions are not necessarily linear, traditional optimization methods may become ineffective and inefficient. Therefore, taking advantage of the special structure of the solution set, they developed a reduction procedure to simplify the problem. They proposed a genetic-based algorithm to find the Pareto optimal solutions.

Khorram and Zarei [63] considered a multiple objective optimization model subject to a system of fuzzy relation equations with max-average composition and presented a reduction procedure in order to reduce problem dimension. They used modified genetic algorithm to solve the problem.

Jiménez et al. [55] considered the problem of solving multiobjective linear programming problems, by assuming that the decision maker has fuzzy goals for each of the objective functions. They showed that, in the case that one of our goals is fully achieved, a fuzzy-efficient solution may not be Pareto-optimal and therefore they proposed a general procedure to obtain a non-dominated solution, which is also fuzzy-efficient.

According to Jiménez and Bilbao [56], in fuzzy optimization it is desirable that all fuzzy solutions under consideration be attainable, so that the decision maker will be able to make a posteriori decisions according to current decision environments. No additional optimization runs will be needed when the decision environment changes or when the decision maker needs to evaluate several decisions to establish the most appropriate ones. In this sense, multiobjective optimization is similar to fuzzy optimization, since it is also desirable to capture the Pareto front composing the solution. The Pareto front in a multiobjective problem can be interpreted as the fuzzy solution for a fuzzy problem. They introduced a multiobjective approach for nonlinear constrained optimization problems.
with fuzzy costs and constraints and a multiobjective evolutionary algorithm to solve the former problem. A case study of a fuzzy optimization problem arising in some import-export companies in the south of Spain was analyzed and the proposed solutions from the evolutionary algorithm considered were given.

Zhang et al. [180] provided an efficient method utilizing a max-pro optimum scheme for solving the max-min decision function in a fuzzy optimization environment. They proposed a method significantly simplifying the max-min optimum solving problem, especially in the case when the number of objectives and constraints is large.

In classical theory, the previous studies on multiobjective optimization using genetic algorithms have been done mainly for problems with a convex feasible domain. In our case, the feasible domain of the problem is generally non-convex and the objective functions considered are not necessarily linear. Therefore, a genetic algorithm is proposed. This raises some issues on establishing and maintaining the Pareto optimal set. With a careful study on the feasible domain, an effective genetic algorithm is proposed in this chapter to obtain the Pareto optimal set without calculating the complete set of minimal solutions of system of fuzzy relation equations.

Max-min composition as used in [78], is conservative in nature; it has limitations over the application towards the real world decision problems. It is generally used when a system requires conservative solutions in the sense that the goodness of one value cannot compensate the badness of another value. In this paper we shall consider the multiobjective optimization problem with fuzzy relation equations composed of max-product composition and will develop a genetic algorithm to determine the satisficing Pareto optimal solution set in the feasible domain of the optimization problem. Other compositions can also be employed depending on the applications. Some outlines for selecting an appropriate composition can be found in Yager [166].
7.4 The problem

Let $A = [a_{ij}]$, $0 \leq a_{ij} \leq 1$, be a $m \times n$ dimensional fuzzy matrix and $b = [b_1, b_2, \ldots, b_n]$, $0 \leq b_j \leq 1$, be a $n$-dimensional vector, then the following system of fuzzy relation equations is defined by $A$ and $b$:

$$x \circ' A = b$$  \hspace{1cm} (7.1)

where $\circ'$ denotes max-$\otimes$ composition of $x$ and $A$. $\otimes$ denotes a compositional operator from product algebra over residuated lattice $L = ([0, 1], \land, \lor, \otimes, \rightarrow, 0, 1)$. It is intended to find a solution vector $x = [x_1, x_2, \ldots, x_m]$, with $0 \leq x_i \leq 1$, such that

$$\max_{i=1}^{m} (x_i \otimes a_{ij}) = b_j, \forall j = 1, 2, \ldots, n$$  \hspace{1cm} (7.2)

Each input of (7.2) may require certain amount of resources which can be considered cost, and a decision maker may wish to achieve certain objectives—a situation which is usually the case for the real-world applications.

Let $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ be the index sets. We are interested in finding the satisficing decisions of the following multiobjective optimization model with max-product fuzzy relation equations as constraints:

$$\text{Min } \{f_1(x), f_2(x), \ldots, f_k(x)\}$$  \hspace{1cm} (7.3)

subject to:

$$\max_{i \in I} (x_i \cdot a_{ij}) = b_j, \forall j \in J$$  \hspace{1cm} (7.4)

$$0 \leq x_i \leq 1, \forall i \in I$$

where $f_k(x)$ is a linear or nonlinear objective function, $k \in K = \{1, 2, \ldots, s\}$. 

175
Let \( X(A,b) = \{x \in [0,1]^n \mid x \circ A = b \} \) be the solution set of fuzzy relation equations (7.4).

For any \( x^1, x^2 \in X \), we say \( x^1 \leq x^2 \) if and only if \( x^1_i \leq x^2_i \), \( \forall i \in I \). Therefore \( \leq \) forms a partial ordering relation on \( X \) and \((X, \leq)\) becomes a lattice. Equations in (7.4) form a system of latticized polynomial equations. \( \bar{x} \in X(A,b) \) is the maximum solution, if \( x \leq \bar{x}, \forall x \in X(A,b) \). Similarly, \( \bar{x} \in X(A,b) \) is a minimal solution, if \( x \leq \bar{x} \) implies \( x = \bar{x}, \forall x \in X(A,b) \). According to [66], if \( X(A,b) \neq \emptyset \), then it is, in general, a non-convex set which can be completely determined by unique maximum solution \( \bar{x} \) and several minimal solutions \( \bar{x} \).

The maximum solution can be computed explicitly by the residual implicator (pseudo complement). The maximum solution of fuzzy relation equations (7.4) can be obtained by assigning

\[
\bar{x} = A \rightarrow b = \left[ \min_{j \in J} (a_{ij} \rightarrow b_j) \right]_{i \in I} \tag{7.5}
\]

where

\[
a_{ij} \rightarrow b_j = \begin{cases} 
1 & \text{if } a_{ij} \leq b_j \\
b_j / a_{ij} & \text{if } a_{ij} > b_j
\end{cases} \tag{7.6}
\]

If \( \bar{X}(A,b) \) denotes the set of all minimal solutions, then the complete solution set of fuzzy relation equations (7.4) can be formed as (7.7).

\[
X(A,b) = \bigcup_{\bar{x} \in \bar{X}(A,b)} \{x \in [0,1]^n \mid \bar{x} \leq x \leq \bar{x}\} \tag{7.7}
\]

Genetic algorithm is proposed for solving problem (7.3)-(7.4). The notion of Pareto optimality is only one consideration in determining the satisficing solutions of multiobjective optimization problems.
The Nobel Prize laureate Herbert A. Simon coined the term “satisficing” to describe the selection of a good enough solution, the selection of a decision that meets a minimum aspiration level or threshold, a threshold, under which solutions are deemed unacceptable, the selection of which occurs in the context of incomplete information or limited computation [142,143]. A satisficing solution may or may not be an optimal economic solution. Therefore, “satisficing” explains the tendency to select the first option that meets a given need or select the option that seems to address most needs rather than the “optimal” solution.

**Definition 7.4.1.** For each \( x \in X(A,b) \), we say \( x \) is a solution vector and define \( z = (f_1(x), f_2(x), \ldots, f_s(x)) \) to be its criterion vector. Moreover, we define \( Z = \{ z \in R^s \mid z = (f_1(x), f_2(x), \ldots, f_s(x)) \text{ for some } x \in X(A,b) \} \).

**Definition 7.4.2.** A point \( \bar{x} \in X(A,b) \) is an efficient or a Pareto optimal solution to the problem (7.3)-(7.4) iff there does not exist any \( x \in X(A,b) \) such that \( f_k(x) \leq f_k(\bar{x}), \forall k \in K \), and \( f_k(x) < f_k(\bar{x}) \) for at least one \( k \). Otherwise, \( \bar{x} \) is an inefficient solution.

**Definition 7.4.3.** For any two criterion vectors, \( z^1, z^2 \in Z \), we say that \( z^1 \) dominates \( z^2 \) iff \( z^1 \leq z^2 \) and \( z^1 \neq z^2 \) i.e. \( z^1_k \leq z^2_k, \forall k \in K \), \( z^1_k < z^2_k \) for at least one \( k \).

**Definition 7.4.4.** \( \bar{z} \in Z \) is said to be non-dominated iff there does not exist any \( z \in Z \) that dominates \( \bar{z} \). Otherwise, \( \bar{z} \) is a dominated criterion vector.

The concept of dominance is applied to the criterion vectors and the concept of efficiency is applied to the solution vectors. From the above definitions it is clear that \( \bar{x} \in X(A,b) \) is efficient or a Pareto optimal solution to the problem (7.3)-(7.4) if its criterion vector is non-dominated in \( Z \). The set of all efficient points is called the efficient set or Pareto optimal set and the set of all non-dominated criterion vectors is called the non-dominated set.
Remark 7.4.5. Major algorithms and procedures of genetic algorithm remain same as that introduced in Chapter 6 except that now notion of Pareto optimality is introduced. For sake of completeness, we introduce them again.

7.5 Initialization

In general, a genetic algorithm initializes the population randomly. Since randomly generated solutions may not be feasible, the proposed genetic algorithm intends to keep the solutions (chromosomes) feasible by randomly generating the individuals inside the feasible domain.

Definition 7.5.1. Given a system of fuzzy relation equations (7.4), a pseudo-characteristic matrix $P = [p_{ij}]_{m \times n}$ is defined as

$$p_{ij} = \begin{cases} 
1 & \text{if } a_{ij} > b_j \\
0 & \text{if } a_{ij} = b_j \\
-1 & \text{if } a_{ij} < b_j
\end{cases}$$

Lemma 7.5.2. For column $j$, if there is only one $i \in I$ such that $p_{ij} = 1$ and $p_{ij'} = -1, \forall i' \neq i$, then $x_i = \bar{x}_i = a_j / a_{ij}$.

Proof. For proof see Chapter 6, Lemma 6.4.2.

Thus from Lemma 7.5.2 it is clear that if there exists some $i \in I$ such that $p_{ij} = 1$ and $p_{ij'} = -1, \forall i' \neq i, i \in I$ then $x_i = \bar{x}_i = b_j / a_{ij}$, i.e., the value of $i$th component has to be fixed.
Algorithm 1: For initialization of the population

1. Get the matrix $A, b$ and size of population, say $u$.
2. Find the maximum solution $\tilde{x}$ by (7.5).
3. If $\tilde{x}' A = b$, continue. Otherwise, stop, the problem is infeasible.
4. For $i = 1, 2, \ldots, m$, if there exists some $i \in I$ such that $p_{ij} = 1$ and $p_{ij} = -1$, $\forall i' \neq i$, $i \in I$ then $x_i = \tilde{x} = x_i = b_j / a_{ij}$; the value of $i$th variable is fixed.
5. Let $\tilde{I}$ denotes the index of the non-fixed variables.
6. Set $k = 1$.
7. WHILE $(k < u)$
   
   For $i = 1, 2, \ldots, \tilde{I}$,
   
   Generate a random number $r$ in the interval $[0, \tilde{x}(i)]$

   Set $k \leftarrow k + 1$

   Output matrix $r$ as the initial population (solution).
8. Check feasibility of each solution. If solution is infeasible, modify it by using Algorithm 2.

Algorithm 2: For maintaining feasibility of solutions

1. Choose a violated constraint $j$. Let $D_j = \{i \in I \mid a_{ij} \geq b_j\}$.
2. Randomly choose an element $k \in D_j$. For $a_{ij} > b_j$ or $\tilde{x}_k = b_j / a_{ij}$, set $x_k = b_j / a_{ij}$.
   
   Otherwise, assign a random number between $[b_j / a_{ij}, \tilde{x}_k]$ to $x_k$.
3. Check the feasibility of the new solution. If the solution is still infeasible, then go to Step 1 and repeat the process. Otherwise, stop.

For initialization procedure, see Example 6.4.3 in Chapter 6.
7.6 Selection/ reproduction procedure

In multiobjective optimization problem, we are dealing with more than one objective functions simultaneously, therefore each individual has a vector of fitness values, instead of a single value. The vector of fitness can be determined by the objective functions. Since genetic algorithms usually require a scalar value of each individual to evaluate its quality, the elements of the fitness need to be combined somehow into a single value [5].

In genetic algorithms, the idea of natural selection that highly fitted individuals will reproduce more often at the cost of lower fitted ones is called reproduction or selection. Reproduction or selection concerns how to select the individuals in the population who will create offspring for the next generation and how many offspring each will create. There are many methods for implementing this, and one commonly used method is Roulette Wheel selection strategy, originally proposed by Holland [52]. The basic idea is to determine selection probability for each individual proportional to the fitness value. It works only for maximization problem. It is not appropriate for minimization problem since it assigns greater probability to the individuals that have a greater fitness value. Another drawback of Roulette Wheel selection strategy is the presence of some super-individuals. A super-individual has a significantly better fitness value and therefore has a comparatively greater probability of selection. As a result, it has a relatively larger number of offsprings and has a tendency to prevent other individuals from being selected and contributing in the next generation. Consequently, after a number of generations, a super-individual may eliminate other individuals and lead to a (local) optimum. To avoid the problem of super-individual, a selection method based on the information of ranks was proposed [88]. In this method, ranks are used for determining the number of offspring to be generated from each individual.

Ranking method for multi-objective optimization utilizes the concept of domination, as explained through Definitions 7.4.1-7.4.4. The rank of each individual, \( r_p \), is determined by the number of dominators or solutions whose criterion vectors dominate this
individual’s criterion vector. It is defined as, \( r_p = \text{pop-size} - \text{number of dominators} \), \( p = 1, 2, \ldots, \text{pop-size} \). The probability of an individual with rank \( r_p \) being selected is

\[
P(\text{Probability of selection of individual with rank } r_p) = \frac{r_p}{\text{pop-size} \sum_{i=1}^{r_p} r_i}
\]

(7.8)

The Roulette wheel method is then applied to reproduce the next generation according to the probability of selection (7.8). Along the run, a set of efficient solutions whose criterion vectors are not dominated by those of other solutions is maintained. We denote \( ES \) as the efficient set. It contains all the efficient solutions obtained so far. \( \overline{ES} \) denote an efficient set of the current generation and is determined relative to the current population. Therefore, the individuals in \( \overline{ES} \) have no dominators with respect to the current population. At each iteration in the selection procedure, after calculating the fitness, we check each individual in \( \overline{ES} \) to find whether its criterion vector can dominate any of the individuals in set \( ES \). At the end, \( ES \) is updated by removing out the dominated individuals from \( ES \) and adding the dominating individuals from \( \overline{ES} \).

7.7 Mutation

In genetic algorithms, mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. The idea of mutation is to slightly modify an individual while maintaining some of the current information in that individual. The purpose of mutation is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.

We know that, with the maximum solution, all components are binding, i.e., there exists some \( j \in J \) such that \( x_i \cdot a_{ij} = b_j \), for all \( i \in I \). Taking the advantage of information contained in the maximum solution, a mutation operator that diversifies the current
population is introduced. Each component of an individual has some chance to be mutated. If a component is selected for mutation operation, its value is reassigned with a random number in the range of 0 and its maximum value. The mutated solution is then checked for feasibility and modified if infeasible. The Algorithm 3 describes the mutation operation.

**Algorithm 3: For mutation operator**

1. For each \( i = 1, 2, \ldots, I \), generate random number \( r_i \in [0,1] \).
2. For \( i = 1, 2, \ldots, I \), if \( r_i \leq \delta \), then randomly assign \( x_i \) a number in the range of \([0, \bar{x}_i]\).
3. Check feasibility of the individual after mutation; if infeasible, then modify it through Algorithm 2.

The parameter \( \delta \) is chosen to be 0.1. This implies that each component of all the solution vectors has 10 percent chances of being mutated. The values of the solution vectors are diversified over the feasible domain avoiding the population to converge to a maximum solution. Feasible mutation operator preserves feasibility of the solutions. Since the relative position of a solution (point) in the feasible domain is changed during mutation by a probability of 0.1, therefore the solution after the mutation operation may become infeasible. Infeasible solution is again projected to feasible domain by applying Algorithm 2.

For mutation procedure, see Example 6.6.1 in Chapter 6.

**7.8 Crossover**

It is well-recognized that the main distinguishing feature of genetic algorithms is the use of crossover. Crossover, also called recombination, is an operator that creates new individuals from the current population. The main role of this operator is to combine pieces of information coming from different individuals in the population. Actually, it recombines genetic material of two parent individuals to create offspring for the next
The basic crossover operation, introduced by Holland [52], is a three-step procedure. First, two individuals are selected at random from the population of parent strings generated by the selection. Second, one or more string locations are selected as crossover points depicting the string segment to exchange. Finally, parent string segments are exchanged and then combined to produce two resulting offspring individuals. The proportion of parent strings undergoing crossover during a generation is controlled by the crossover rate \( p_c \in [0,1] \), which determines how frequently the crossover operator is invoked.

There are many crossover operators in the literature proposed for non-constrained or convex optimization problems [53]. To preserve feasibility of solutions after the crossover operation, one might consider some linear combination of two individuals. However, since the feasible domain of fuzzy relation equations is non-convex, the linear combinations of two feasible individuals will very likely results in an infeasible one.

**Definition 7.8.1.** Given a connected set \( S \) and any two points \( x^1, x^2 \) of \( X \), \( 0 \leq \lambda \leq 1 \), \( \gamma \geq 1 \),

(i) A linear contraction of \( x^1 \) supervised by \( x^2 \) is defined by

\[
x^1 \leftarrow \lambda x^1 + (1-\lambda)x^2
\]

(ii) A linear extraction of \( x^1 \) supervised by \( x^2 \) is defined by

\[
x^1 \leftarrow \gamma x^1 - (\gamma-1)x^2
\]

The parameter \( \lambda \) and \( \gamma \) are set to be some positive numbers close to 1. It means that the step lengths of linear contraction and extraction supervised by another parent are very small. It is necessary for preserving the feasibility of the solution since the large movement may pull the solution point out of the feasible domain. By the crossover of two parents, two new individuals are created, one by contraction (or extraction) of \( x^1 \) supervised by \( x^2 \) and other by contraction (or extraction) of \( x^2 \) supervised by \( x^1 \). The
two new individuals obtained after the crossovers have probability $\varepsilon$ of contraction (or extraction) with the maximum solution $\tilde{x}$. Thus crossover is a three-point crossover operator. The three-point crossover operations on a parent will be both supervised by a maximum solution and supervised by another parent. Crossover operator is explained in the Algorithm 4.

**Algorithm 4: For crossover operator**

1. Set the parameters; $\theta = 0.5$ (equal probability of performing a linear contraction and extraction), $\xi = 0.1$, $\lambda = 0.995$, $\gamma = 1.005$ (step lengths of linear contraction and extraction supervised by another parent are very small for maintaining feasibility of solutions).

2. Randomly select two different solutions, say, $x^1$ and $x^2$ from the current population.

3. Generate a random number $\tau_1 \in [0,1]$.
   - If ($\tau_1 \geq 0.5$)
     \[ x_{new}^1 \leftarrow \gamma x^1 - (\gamma - 1)x^2 \]
   - Else
     \[ x_{new}^1 \leftarrow \lambda x^1 + (1 - \lambda)x^2 \]

4. Generate a random number $\tau_2 \in [0,1]$.
   - If ($\tau_2 \geq 0.5$)
     \[ x_{new}^2 \leftarrow \gamma x^2 - (\gamma - 1)x^1 \]
   - Else
     \[ x_{new}^2 \leftarrow \lambda x^2 + (1 - \lambda)x^1 \]

5. Generate a random number $\tau_3 \in [0,1]$.
   - If ($\tau_3 < \xi$)
\{ Generate a random number $r_4 \in [0,1]$ \\
If ($r_4 < 0.5$) \\
\hspace{1cm} x_{\text{New}}^1 \leftarrow \lambda x_{\text{New}}^1 + (1-\lambda)\bar{x} \\
Else \\
\hspace{1cm} x_{\text{New}}^1 \leftarrow \gamma x_{\text{New}}^1 - (\gamma-1)\bar{x} \\
\} \\
Else \\
\hspace{1cm} Go to Step 6. \\

6. Generate a random number $r_5 \in [0,1]$. \\
If ($r_5 < \xi$) \\
\hspace{1cm} \{ Generate a random number $r_6 \in [0,1]$. \\
\hspace{2cm} If ($r_6 < 0.5$) \\
\hspace{3cm} x_{\text{New}}^2 \leftarrow \lambda x_{\text{New}}^2 + (1-\lambda)\bar{x} \\
\hspace{3cm} Else \\
\hspace{4cm} x_{\text{New}}^2 \leftarrow \gamma x_{\text{New}}^2 - (\gamma-1)\bar{x} \\
\hspace{1cm} \} \\
Else \\
\hspace{1cm} Go to Step 7. \\

7. Check $x_{\text{New}}^1$, $x_{\text{New}}^2$, $\forall i = 1, 2, \ldots, m$. If for any $i$, $x_{\text{New}}^1$ (or $x_{\text{New}}^2$) lies out of range of $[0,\bar{x}_i]$, then assign $x_{\text{New}}^1$ (or $x_{\text{New}}^2$) a random number between $[0,\bar{x}_i]$. \\

8. Check feasibility of both solutions. If infeasible, then modify them by using Algorithm 2. \\

For crossover procedure, see Example 6.7.3 in Chapter 6.
7.9 Hill climbing method

Hill climbing method, also known as local improvement method is used in improving the solutions obtained through genetic algorithm. It uses the iterative improvement technique; the technique is applied to a single point (the current point) in the search space. During a single iteration, a new point is selected from the neighborhood of the current point. If the new point provides a better value of the objective function, the new point becomes the current point. Otherwise, some other neighbor is selected and tested against the current point. The method terminates if no further improvement is possible. It is clear that the hill climbing methods provide local optimum values only and these values depend on the selection of the starting point. Moreover, there is no information available on the relative error (with respect to the global optimum) of the solution found. To increase the chances to succeed, hill climbing methods usually are executed for a (large) number of different starting points (these points need not be selected randomly - a selection of a starting point for a single execution may depend on the result of the previous runs).

In each iteration, we perturb the solutions in the efficient set $ES$ by a small diameter, say 0.1. First, for each solution in $ES$, a vector containing $m$ random numbers in the range of $[-0.05, 0.05]$ is generated, and added to the solution. Then the feasibility of the new solution is checked and if it is infeasible, then we make it feasible using Algorithm 2. Then we check that if this new solution can dominate some solutions in $ES$. If yes, then the dominated solutions are eliminated from $ES$, and the new solution (dominating solution) is added to $ES$. If no domination occurs, then new solution is not added to $ES$, since hill climbing method searches for a better solution, not an equally good one. In the same way we perturb each solution of $ES$. The hill climbing method is performed 100 times for each individual.
7.10 Construction of fuzzy multiobjective optimization problems

To obtain the measurement of performance of propose genetic algorithm, test problems are considered by taking multiobjective objective functions and constructing a feasible problem by randomly generating a fuzzy matrix $A$ and constructing vector $b$ from $A$ according to some criteria. The solution set $X(A,b)$ of fuzzy relation equations constructed by this method is not empty.

Randomly generated fuzzy relation equations may not be feasible. The procedure of generating feasible max-product fuzzy relation equations is designed in Algorithm 5.

Algorithm 5: Randomly generating feasible max-product fuzzy relation equations

1. Generate a $m \times n$ matrix $A$ whose elements are random numbers from $[0,1]$.

2. Generate vector $b$ such that its $j$th element is a random number from $[a_j^q, a_{\text{max}}^q]$, where $a_{\text{max}} = \max_{i \in I} (a_{ij})$ and $a_j^q = 0.75 (a_{max})$.

3. For ($i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$)
   
   {For ($a_{ij} > b_j$)
   
   {For ($k \neq j$)
   
   {If ($b_k / a_{ik} < b_j / a_{ij}$ & $a_{ik} > b_k$)
   
   Set $a_{ik} = (b_k \times a_{ij}) / b_j$
   
   }
   
   }
   
   }

The random fuzzy relation equations generated by this algorithm have feasible solutions.
Algorithm 6: Summary of the proposed method

1. Initialize maximum number of generations, $Gen = 5000$.
2. Set the current generation number $c = 1$.
3. Generate initial population using Algorithm 1.
4. Perform selection procedure and establish the efficient set $ES$ by putting all efficient solutions of the initial population into it.
5. Perform mutation procedure using Algorithm 3.
7. Again apply selection/reproduction procedure and update the efficient set $ES$.
8. While $c < Gen$, set $c \leftarrow c + 1$. Go to step 4.

7.11 Illustrations

This section shows the implementation of the proposed genetic algorithm for multiobjective optimization problem. We will consider multiobjective optimization problem with linear objective functions and nonlinear objective functions. It is shown that the proposed genetic algorithm is capable of finding efficient Pareto optimal set of the problem.

**Linear objective functions:** Consider multiobjective linear optimization problems and randomly generated fuzzy relation equations with max-product composition to investigate the nature of the Pareto optimal solution set.

**Example 7.11.1.** Consider a four dimensional problem with randomly generated fuzzy matrices $A$ and $b$ as follows:
\[ A = \begin{bmatrix} 0.0986 & 0.8471 & 0.5250 & 0.2460 \\ 0.1197 & 0.5436 & 0.8444 & 0.9309 \\ 0.5695 & 0.7093 & 0.0083 & 0.9550 \\ 0.5333 & 0.5834 & 0.6204 & 0.8944 \end{bmatrix} \]

\[ b = [0.5139 \quad 0.8265 \quad 0.7817 \quad 0.8618] \]

The maximum solution is obtained as \([0.9757 \quad 0.9258 \quad 0.9024 \quad 0.9636]\). For this particular problem, the values of \(x_1\) and \(x_2\) of all solution vectors have to be fixed at 0.9757 and 0.9258, respectively. Therefore, we can focus on \(x_3\) and \(x_4\) only. Since the problem is reduced as a two-dimensional problem, the result can be presented graphically.

The test results for some multiple linear optimization problems with this system of fuzzy relation equations as constraints are discussed below.

**Case 1.** Min \( \begin{bmatrix} f_1(x) = -0.6x_1 + 0.5x_2 + 0.1x_3 + 0.3x_4, \\ f_2(x) = 0.8x_1 - 0.4x_2 + 0.2x_3 - 0.3x_4. \end{bmatrix} \)

The Pareto optimal solution set for the above problem is obtained as \( \{x_1 = 0.9757, x_2 = 0.9258, x_3 = 0, x_4 = 0.9636\} \cup \{x_1 = 0.9757, x_2 = 0.9258, x_3 = 0.9024, 0 \leq x_4 \leq 0.6600\} \).

Figure 7.1 shows the Pareto optimal solutions obtained without local improvement. Figure 7.2 shows Pareto optimal solutions obtained with local improvement. Figure 7.3 shows the plot of objective values against the set of efficient solutions.
Figure 7.1: Pareto optimal solutions-Example 7.11.1-Case 1 without local improvement

Figure 7.2: Pareto optimal solutions-Example 7.11.1-Case 1 with local improvement

Figure 7.3: Pareto front-Example 7.11.1-Case 1
Case 2. Min \[ \begin{align*} f_1(x) &= -0.6x_1 + 0.5x_2 - 0.1x_3 - 0.3x_4, \\ f_2(x) &= 0.8x_1 - 0.4x_2 - 0.2x_3 - 0.3x_4. \end{align*} \]

The Pareto optimal solution set for the above problem is obtained as \( \{ x_1 = 0.9757, x_2 = 0.9258, x_3 = 0.9024, x_4 = 0.9636 \} \) which is also the maximum solution. Thus this problem has only one Pareto optimal solution as shown in Figure 7.4.

![Figure 7.4: Pareto optimal solutions-Example 7.11.1-Case 2](image)

Case 3. Min \[ \begin{align*} f_1(x) &= -0.6x_1 + 0.5x_2 - 0.1x_3 + 0.3x_4, \\ f_2(x) &= 0.8x_1 - 0.4x_2 + 0.2x_3 - 0.3x_4. \end{align*} \]

The Pareto optimal solution set for the above problem is obtained as \( \{ x_1 = 0.9757, x_2 = 0.9258, 0 \leq x_3 \leq 0.9024, x_4 = 0.9636 \}, \{ x_1 = 0.9757, x_2 = 0.9258, x_3 = 0.9024, 0 \leq x_4 \leq 0.9636 \} \) shown in the Figure 7.5.
The semi-positive polar cones [149] generated by the negative gradients of two objective functions $f_1$ and $f_2$ for three cases are shown in Figures 7.6-7.8.
Figure 7.7: The semi-positive polar cone-Example 7.11.1-Case 2

Figure 7.8: The semi-positive polar cone-Example 7.11.1-Case 3
$v'$ and $v^2$ correspond to the negative gradients of the first objective and the second objective, respectively. $y'$ and $y^2$ are the vectors that are perpendicular to $v'$ and $v^2$, correspondingly. The shaded area in each case represents the semi-positive cone or the domination set, i.e., the set of all nondomination criterion vectors, with respect to the extreme point (vertex) of the cone. Any point in this set dominates the vertex of the cone. Therefore, these cones can be used to indicate the efficient solutions of each case.

**Example 7.11.2.** Consider a five dimensional problem with randomly generated fuzzy matrices $A$ and $b$ as follows:

$$
A = \begin{bmatrix}
0.0820 & 0.4985 & 0.9396 & 0.2262 & 0.2143 \\
0.1538 & 0.7378 & 0.2107 & 0.9325 & 0.7379 \\
0.2558 & 0.3074 & 0.9563 & 0.7426 & 0.6280 \\
0.1048 & 0.7715 & 0.6356 & 0.5133 & 0.0907 \\
0.2852 & 0.2026 & 0.4252 & 0.5417 & 0.8121 \\
\end{bmatrix}
$$

$$
b = [0.2542 0.7239 0.9505 0.9149 0.7240]
$$

The maximum solution is obtained as $[1.0000 0.9811 0.9940 0.9383 0.8915]$. For this particular problem, the values of $x_2$ and $x_3$ of all solution vectors have to be fixed at 0.9811 and 0.9940, respectively. Therefore, we can focus on $x_1$, $x_4$ and $x_5$ only. The result can be presented graphically. The test results for an optimization problem with three linear objective functions and this system of fuzzy relation equations as constraints are discussed below.

$$
\begin{align*}
\text{Min } f_1(x) &= 5x_1 + 3x_2 - 4x_3 + x_4 - x_5; \\
\text{Min } f_2(x) &= 3x_1 + x_2 + 10x_3 - x_4 + x_5; \\
\text{Min } f_3(x) &= x_1 + x_2 - x_3 + 2x_4 + x_5.
\end{align*}
$$

The Pareto optimal solution set for the above problem is obtained as $(x_1 = 0, x_2 = 0.9811, x_3 = 0.9940, 0 \leq x_4 \leq 0.9383, x_5 = 0.8915)$, $\cup (x_1 = 0, x_2 = 0.9811, x_3 = 0.9940, 0 \leq x_4 \leq 0.9383, x_5 = 0.8915)$. 

194
$x_3 = 0.9940, x_4 = 0.9383, 0 \leq x_5 \leq 0.8915 \cup \{x_1 = 0, x_2 = 0.9811, x_3 = 0.9940, x_4 = 0, x_5 = 0\} \\
\cup \{x_1 = 0, x_2 = 0.9811, x_3 = 0.9940, 0 \leq x_4 \leq 0.9383, 0 \leq x_5 \leq 0.8915, x_4 \neq x_5\}$ shown in Figure 7.9.

![Figure 7.9: Pareto optimal solutions-Example 7.11.2](image)

**Nonlinear objective functions:** We consider multi-objective nonlinear optimization problems and randomly generated fuzzy relation equations with max-product composition to investigate the nature of the Pareto optimal solution set.

**Example 7.11.3.** Consider a three dimensional problem with randomly generated fuzzy matrices $A$ and $b$ as follows:

$$A = \begin{bmatrix} 0.9631 & 0.6416 & 0.3580 \\ 0.4869 & 0.3063 & 0.9382 \\ 0.8175 & 0.6609 & 0.4877 \end{bmatrix}$$

$$b = [0.8985 \ 0.5985 \ 0.8781]$$

195
The maximum solution is obtained as [0.9329  0.9359  0.9056]. For this particular problem, the value of $x_1$ and $x_2$ of all solution vectors has to be fixed at 0.9329 and 0.9359 respectively. Therefore, we can focus on $x_3$ only. The result can be presented graphically. The test results for an optimization problem with two linear objective functions and this system of fuzzy relation equations as constraints are discussed below.

\[
\begin{bmatrix}
  f_1(x) = -25x\sin(10x_3 + 10), \\
  f_2(x) = 10([x_3 / 0.4] + 1) + 18x_3,
\end{bmatrix}
\]

The Pareto optimal solution set for the above problem is obtained as \{ $x_1 = 0.9329$, $x_2 = 0.9359$, $x_3 = 0$ \} $\cup$ \{ $x_1 = 0.9329$, $x_2 = 0.9359$, $0.1992 \leq x_3 \leq 0.4137$ \}. Figure 7.10 shows the plot of the objective functions $f_1$ and $f_2$ against $x_3$ and the Pareto front. Figure 7.11 shows the Pareto optimal solution set.

![Figure 7.10: Plot of the function and Pareto front-Example 7.11.3](image)
Example 7.11.4. Consider a three dimensional problem with randomly generated fuzzy matrices $A$ and $b$ as follows:

$$A = \begin{bmatrix}
0.7085 & 0.1680 & 0.6862 & 0.7176 & 0.0235 \\
0.7428 & 0.3720 & 0.1631 & 0.6134 & 0.4239 \\
0.7134 & 0.3647 & 0.1901 & 0.4624 & 0.0142 \\
0.5559 & 0.7804 & 0.7153 & 0.8169 & 0.6494
\end{bmatrix}$$

$$b = [0.7024 \ 0.7422 \ 0.6803 \ 0.7769 \ 0.6176]$$

The maximum solution is obtained as $[0.9914 \ 0.9456 \ 0.9846 \ 0.9511]$. For this particular problem, the value of $x_4$ of all solution vectors has to be fixed at 0.9511. Therefore, we can focus on $x_1$, $x_2$ and $x_3$ only. The result can be presented graphically. The test results for an optimization problem with two nonlinear objective functions and this system of fuzzy relation equations as constraints are discussed below.
Example 7.11.5. Consider a four dimensional problem with randomly generated fuzzy matrices $A$ and $b$ as follows:

$$A = \begin{bmatrix}
0.6041 & 0.4209 & 0.3220 & 0.9373 \\
0.0659 & 0.0633 & 0.0200 & 0.5058 \\
0.9189 & 0.0478 & 0.4578 & 0.7877 \\
0.1072 & 0.4944 & 0.5098 & 0.5556
\end{bmatrix}$$

$$b = [0.8656 \ 0.4182 \ 0.4312 \ 0.9314]$$
The maximum solution is obtained as \([0.9937 \ 1.0000 \ 0.9420 \ 0.8459]\). For this particular problem, the value of \(x_1\) and \(x_3\) of all solution vectors has to be fixed at 0.9937 and 0.9420 respectively. Therefore, we can focus on \(x_2\) and \(x_4\) only. The result can be presented graphically.

The test results for an optimization problem with two nonlinear objective functions and this system of fuzzy relation equations as constraints are discussed below.

\[
\begin{align*}
\min f_1(x) &= 10(x_2 - 0.5)^2 + 10(x_4 - 0.5)^2 + 5, \\
\min f_2(x) &= 10(x_2 - 0.7)^2 + 10(x_4 - 0.7)^2 + 5.
\end{align*}
\]

The Pareto optimal solution set for the above problem is obtained as \(\{x_1 = 0.9937, \ 0.5 \leq x_2 \leq 0.7, \ x_3 = 0.9420, \ x_4 = x_2\}\). Figure 7.13 represents the surfaces of the two objective functions and Figure 7.14 shows the contour plots of the solutions yielding the objective value 5.4, 5.8, and 6.2 referencing from the inner to outer contour. Value of first objective is minimum at point \(E\) and value of second objective is minimum at point \(F\). If we consider points \(E\) and \(G\), then both of them lies on the same contour with respect to the second objective function. Therefore, both points \(E\) and \(G\), gives the same value of the second objective function. Also, point \(E\) gives the smaller value of the first objective function than point \(G\). Therefore, \(E\) dominates \(G\). Similarly, we can find that any points lying between \(E\) and \(F\) are Pareto optimal solutions.

Figure 7.15 shows the Pareto optimal solution set without local improvement and Figure 7.16 shows the Pareto optimal solution set with local improvement.
Figure 7.13: Plot of function-Example 7.11.5

Figure 7.14: Contour plots-Example 7.11.5

Figure 7.15: Pareto optimal solutions-Example 7.11.5 without local improvement
7.12 Conclusion

This chapter discusses a multiobjective optimization problem with max-product fuzzy relation equations as constraints. We have proposed a genetic algorithm for solving such problem. The problem is simplified by finding out the fixed components for all the solutions. The feasibility of solutions is always maintained during the crossover and mutation operations. The efficient set is constantly updated and finally hill climbing method is employed to further improve the final efficient set. Pareto optimal set of the problem may be a single solution, it may be located on one edge or on two edges, it may be located at extreme points or some interior points. The proposed genetic algorithm is capable of determining the satisficing or Pareto optimal solutions set in all the cases for both linear and nonlinear problems. This shows that the proposed algorithm is effective in searching all over the solution space. Computational time of the algorithm was found proportional to the size of the Pareto optimal set. Efficient sets in two and three-dimensional spaces are presented graphically.