Applied Mathematics, 2014, 5, 3297-3310
Published Online December 2014 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2014.521307

Three-Objective Programming with Continuous Variable Genetic Algorithm

Adugna Fita

Department of Mathematics, Adama Science and Technology University, Adama, Ethiopia
Email: fitaadu@yahoo.com

Received 28 September 2014; revised 20 October 2014; accepted 6 November 2014

Copyright © 2014 by author and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

The subject area of multiobjective optimization deals with the investigation of optimization problems that possess more than one objective function. Usually, there does not exist a single solution that optimizes all functions simultaneously; quite the contrary, we have solution set that is called nondominated set and elements of this set are usually infinite. It is from this set decision made by taking elements of nondominated set as alternatives, which is given by analysts. Since it is important for the decision maker to obtain as much information as possible about this set, our research objective is to determine a well-defined and meaningful approximation of the solution set for linear and nonlinear three objective optimization problems. In this paper a continuous variable genetic algorithm is used to find approximate near optimal solution set. Objective functions are considered as fitness function without modification. Initial solution was generated within box constraint and solutions will be kept in feasible region during mutation and recombination.

Keywords

Chromosome, Crossover, Heuristics, Mutation, Optimization, Population, Ranking, Genetic Algorithms, Multi-Objective, Pareto Optimal Solutions, Parent Selection

1. Introduction and Backgrounds

Three-objectives programming problem is special cases of multicriteria optimization problem with three objectives. Multiobjective optimization examines problems featuring several different objective functions which have to be considered simultaneously. Usually these objectives are also competing with each other which aggravates the situation. For instance, a fundamental challenge in portfolio management is to maximize the rate of return while reducing the risk involved in the investment(s) at the same time. In general there does not exist a single solution that optimizes all functions of the problem at once. Thus, a selection of alternatives, so-called nondominated set and elements of this set are usually infinite. It is from this set decision made by taking elements of nondominated set as alternatives, which is given by analysts. But practically extraction of nondominated solutions and setting fitness function are difficult.

Typically, the set of efficient solutions is very large and the different solutions are incomparable with each other with respect to the employed ordering concept. The reason for the incomparability is that we utilize partial orders in vector spaces rather than total orders.

In order to solve our decision making problem by some systems analytical methods, we usually require that degrees of objectives be represented in numerical terms, which may be of multiple kinds even for one objective. In order to exclude subjective value judgment at this stage, we restrict these numerical terms to physical measures (for example money, weight, length, and time).

As a performance index, for the objective Y_i an objective function $f_i : X \to \mathbb{R}^1$ is introduced. Where R^1 denotes one dimensional Euclidean space. The value $f_i(x)$ indicates how much impact is given on objective Y_i by performing an alternative x. In this paper we assume that a smaller value for each objective function is preferred to large one [1].

Now we can formulate our decision making problems as a Multiobjective optimization problem:

$$\begin{matrix} f_1(x) \to \text{Min/Max} \\ f_2(x) \to \text{Min/Max} \\ \vdots \\ f_p(x) \to \text{Min/Max} \end{matrix} \quad \text{Over, } x \in X. \tag{1.1}$$

where:

$$X = \{x \in R^n : g_j(x) \le 0, j = 1, 2, \cdots, m, x \ge 0\} \tag{1.2}$$

$$Y = \{y \in R^p : y_1 = f_1(x), y_2 = f_2(x), \cdots, y_p = f_p(x), x \in X\} \tag{1.3}$$

In some cases, some of the objective functions are required to be maintained under given levels prior to minimizing other objective functions. Denoting these objective functions by $g_j(x)$, we require that

$$g_j(x) \le 0, \quad j = 1, 2, \cdots, m \tag{1.4}$$

Such a function $g_j(x)$ is generally called a constraint function. According to the situation, we consider either the problem (1.1) itself or (1.1) accompanied by the constraint conditions (1.4).

Of course, an equality constraint $h_k(x) = 0$ can be embedded within two inequalities $h_k(x) \le 0$ and $-h_k(x) \le 0$, and hence, it does not appear explicitly in this paper.

We call the set of alternatives X the feasible set of the problem. The space containing the feasible set is said to be a decision space, whereas the space that contains the image of the feasible set $Y = f(X)$ is referred to as criterion space [2].

Unlike the traditional mathematical programming with a single objective function, an optimal solution in the sense of one that minimizes all the objective function simultaneously does not necessarily exist in multiobjective optimization problem, and hence, we are in trouble of conflicts among objectives in decision making; the final decision should be made by taking the total balance of objectives into account [3].

Scalarization approach is one of the solution methods in multiobjective programming that is studied by different scholars. In solving scalarized problems which are obtained by weighting sum of objective functions to single objective, each substitute problem is characterized by a unique weighting parameter vector, the entries of which are the specific weights for the individual objective functions [1] [4]. A *fundamental challenge lies in the selection of the* "*right*" *weighting parameters* that lead to different optimal solutions that subsequently constitute meaningful approximations of the efficient sets.

Vector evaluated genetic algorithm (VEGA) Schaffer (1985) presents one of the first treatments of multi-objective genetic algorithms, although he only considers unconstrained problems [5]. The general idea behind Schaffer's approach, called the vector evaluated genetic algorithm (VEGA), involves producing smaller subsets of the original population, or sub-populations, within a given generation, [6] [7]. One sub-population is created by evaluating one objective function at a time rather than aggregating all of the functions.

The process is based on the idea that the minimum of a single objective function is a Pareto optimal point (assuming the minimum is unique). However, considering only one objective function at a time is comparable to setting all but one of the weights to zero, Goldberg (1989), and Fonseca and Fleming (1993) provide detailed explanations and critiques of Schaffer's ideas. A class of alternatives to VEGA involves giving each member of a population a rank based on whether or not it is dominated (Goldberg 1989; Fonseca and Fleming 1993); Fitness then is based on a design's rank within a population.

Ishibuchi and Murata (1996), and Murata *et al.* (1996) use a procedure called an elitist strategy, which functions independent of rank. As with the Pareto-set filter, two sets of solutions are stored: a current population and a tentative set of non-dominated solutions, which is an approximate Pareto set [8]. With each generation, all points in the current population that are not dominated by any points in the tentative set are added to the set and dominated points in the set are discarded. After crossover and mutation operations are applied, a user's specified number of points from the tentative set is reintroduced into the current population. These points are called elite points. In addition, the k solutions with the best values for each objective function can be regarded as elite points and preserved for the next generation (Murata *et al.* 1996).

Two points, called candidate points, are randomly selected from the current population and compute for survival in the next generation [9]. A separate set of points called a tournament set or comparison set is also randomly compiled. The candidate points are then compared with each member of the tournament set.

If there is only one candidate that is non-dominated relative to the tournament set, that candidate is selected to be in the next generation. However, if there is no preference between candidates, or when there is a tie, fitness sharing is used to select a candidate. The user specifies the size of the tournament set as a percentage of the total population.

Consequently, the size of the tournament set imposes the degree of difficulty in surviving, which is called the domination pressure. An insufficient number of Pareto optimal points will be found if the tournament size is too small, and premature convergence may result if the tournament size is too large [7] [10].

Fitness sharing is a common niche technique the basic idea of which is to penalize the fitness of points in crowded areas, thus reducing the probability of their survival to the next generation [11]; Deb 1989; Srinivas and [12]. The fitness of a given point is divided by a constant that is proportional to the number of other points within a specified distance in the criterion space. In this way, the fitness of all the points in a niche is shared in some sense, thus the term "fitness sharing" [7].

Different techniques are discussed that serve as potential issues in a genetic multi-objective optimization algorithm. In accordance with much of the literature on multi-objective genetic algorithms, constraints are not addressed directly. It is assumed that a penalty approach is used to treat constraints and external penalty function is used in this paper as seen in **Figure 1**. But those approaches are difficult to model the fitness function that best evaluates the solution (chromosomes) for further selections and keeping those solutions in feasible region during mutation and recombination [13].

In this paper objective function and variables are taken without modification and continuous variable genetic algorithm is used. Variables are considered in box constraint and initial solution will be generated within box constraint and will keep in feasible region during mutation and recombination.

2. Preference Orders and Domination Structures

Preference Orders represents the preference attitude of the decision maker in the objective space. Ranking of objectives with multiple attributes or solving the multi-dimensional problem

It is usually represented by a binary relation. For a given pair, y^1 and y^2:

- y^1 is preferred to y^2, denoted by $y^1 > y^2$.
- y^1 is less preferred to y^2, denote by $y^1 < y^2$.
- y^1 is equally preferred to y^2, denoted $y^1 \sim y^2$.

The preference relation between y^1 and y^2 is no related, denote by $y^1 \| y^2$.

Since ordering in objective space is incomplete (*i.e.* a partial ordering) which are not directly comparable with each other the problem has more than one solution there does not necessarily exist a solution that is best with respect to all objectives because of conflict among objectives [1] [3] [14].

There usually exist a set of solutions; nondominated or efficient or Pareto optimal solutions. An alternative is Pareto optimal or non-dominated, if it is: best in at least one criterion (better than any other alternative); or equal to the best in at least one criterion without being worse in all other criteria. For minimization problem decision vector X_1 is preferred to X_2:

If $f_i(X_1) \le f_i(X_2)$ and $f_i(X_1) < f_i(X_2)$ for at least one i, for $i = 1, 2, \cdots, m$. [15]

Pareto optimal solutions form (possibly nonconvex and non-connected) Pareto optimal set as seen in **Figure 1**, **Figure 2**. But for linear problems where the objective and constraint are linear the Pareto optimal solutions set is convex as seen in **Figure 3**. In the case of **Figure 2**, **Figure 4** the non dominated red region is not necessarily convex as the problem are non linear.

In [1] Pareto optimal solution(s) exist if the objective functions are lower semi continuous and the feasible region is nonempty and compact. In some literatures, non dominated set of solution are found on closure (boundary) of the feasible surface which is clearly seen in **Figures 4-6**.

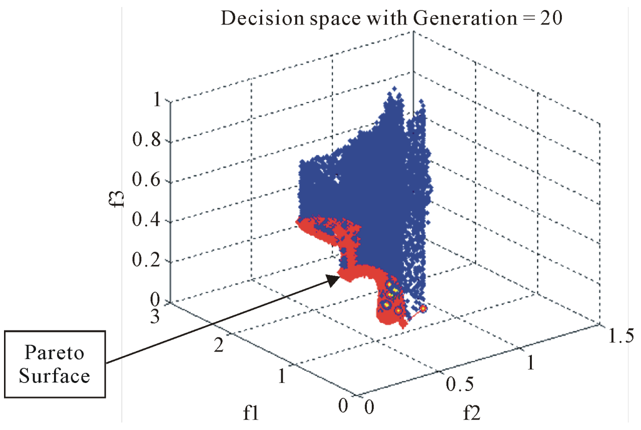

Figure 1. Simulation results.

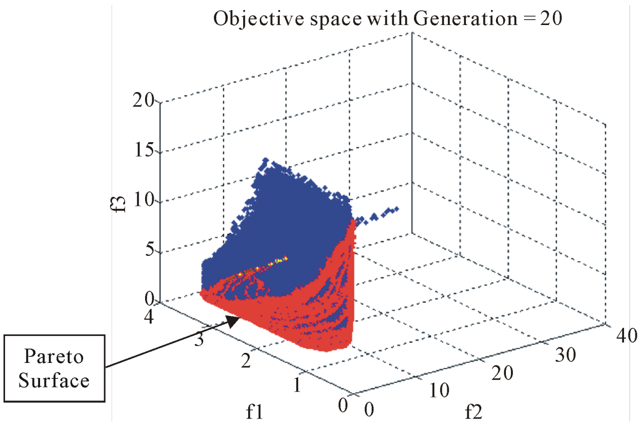

Figure 2. Simulation results.

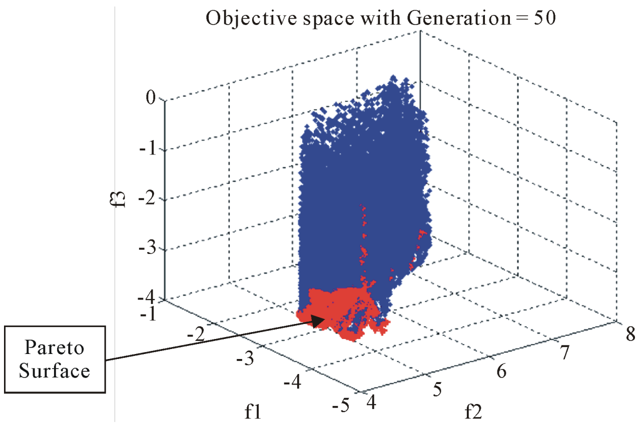

Figure 3. Simulation results.

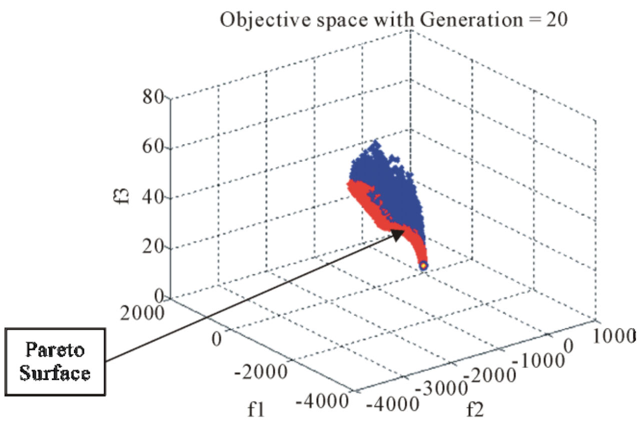

Figure 4. Simulation results.

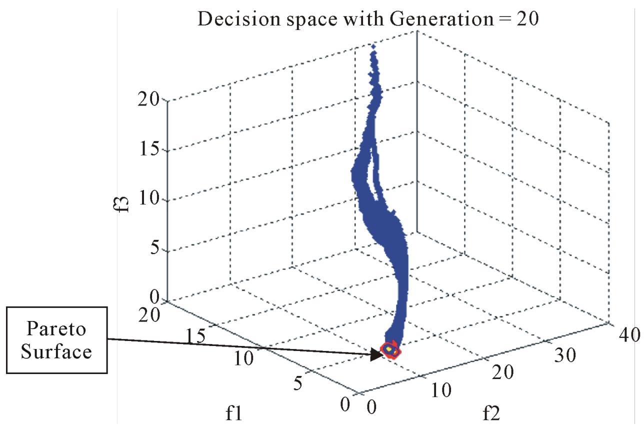

Figure 5. Simulation results.

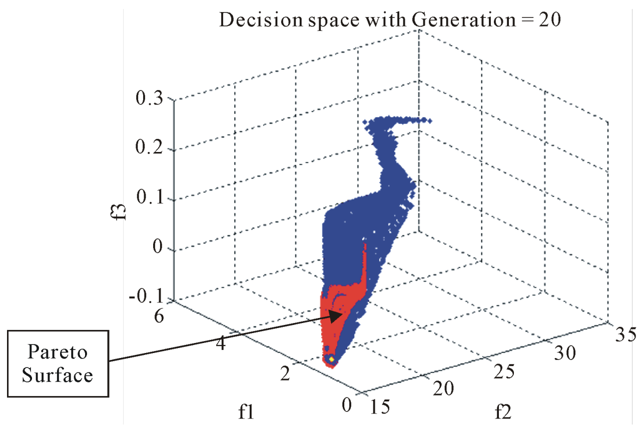

Figure 6. Simulation results.

Almost in all Figures the red surface is nondominated solutions while the blue region is feasible solution but dominated by objective values on red surfaces.

Even though, the existence of efficient solution is mathematically proved under some conditions such as the convexity, connectedness, and compactness of feasible region and lower semi continuity of objectives functions. But practical estimation of efficient set is very difficult [13].

Therefore, in the next chapter one of a global search heuristics which find the true or approximate of near optimal solutions to problems is discussed. This method is less susceptible to the connectivity and convexity of feasible set (search space) **Figure 1**, **Figure 6**. In addition to this continuity and differentiability of objective functions does not require.

3. Continuous Variable Genetic Algorithm

3.1. Introduction

Genetic algorithms is a class of probabilistic optimization algorithms inspired by the biological evolution process uses concepts of "Natural Selection" and "Genetic Inheritance" (Darwin 1859) originally developed by John Holland [16] [17]. Particularly well suited for hard problems where little is known about the underlying search space.

The specific mechanics of the algorithms involve the language of microbiology and, in developing new potential solutions, through genetic operations. A population represents a group of potential solution points. A generation represents an algorithmic iteration. A chromosome is comparable to a design point, and a gene is comparable to a component of the design vector. Genetic algorithms are theoretically and empirically proven to provide robust search in complex phases with the above features.

Many search techniques required auxiliary information in order to work properly. For e.g. Gradient techniques need derivative in order to chain the current peak and other procedures like greedy technique requires access to most tabular parameters whereas genetic algorithms do not require all these auxiliary information. GA is blind to perform an effective search for better and better structures they only require objective function values associated with the individual strings.

A genetic algorithm (or GA) is categorized as global search heuristics used in computing to find true or approximate solutions to optimization problems.

3.2. Initialization of GA with Real Valued Variables

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions.

Commonly, the population is generated randomly, covering the entire range of possible solutions (the search space).

To begin the GA, we define an initial population of N_{pop} chromosomes. A matrix represents the population with each row in the matrix being a $1 \times N_{var}$ array (chromosome) of continuous values. Given an initial population of N_{pop} chromosomes, the full matrix of $N_{pop} \times N_{var}$ random values is generated by:

$$pop = rand\left(N_{pop}, N_{var}\right)$$

If variables are normalized to have values between 0 and 1, If not the range of values is between X_{lb} and X_{ub}, then the unnormalized values are given by:

$$X = \left(X_{ub} - X_{lb}\right) X_{norm} + X_{lb}$$

where X_{lb}: lowest number in the variable range;

X_{ub}: highest number in the variable range;

X_{norm}: normalized value of variable [17].

But one can generate an initial feasible solution from search space simply as,

```
for $i = 1 : N_{pop}$
 $pop(i,:) = \left(X_{ub} - X_{lb}\right) \cdot rand(1, nvar) + X_{lb};$   % generate real valued population matrix
end; this generates $N_{pop} \times N_{var}$ population matrix.
```

3.3. Evaluation (Fitness Function)

Fitness values are derived from the objective function values through a scaling or ranking function. Note that for multiobjective functions, the fitness of a particular individual is a function of a vector of objective function values. Multiobjective problems are characterized by having no single unique solution, but a family of equally fit solutions with different values of decision variables.

Therefore, care should be taken to adopt some mechanism to ensure that the population is able to evolve the set of Pareto optimal solutions.

Fitness function measures the goodness of the individual, expressed as the probability that the organism will live another cycle (generation). It is also the basis for the natural selection simulation better solutions have a better chance to be the next generation. In many GA algorithm fitness function is modeled according to the problem, but in this paper we use objective functions as fitness function.

$$\text{Cost} = f\left(\text{chromosome}\right) \quad \text{that means}$$

$$\text{Cost1} = f\left(X_1 \quad X_2 \quad X_3 \quad X_4 \quad \cdots \quad X_{nvar}\right)$$

$$\text{Cost2} = f\left(X_1 \quad X_2 \quad X_3 \quad X_4 \quad \cdots \quad X_{nvar}\right)$$

$$\text{Cost3} = f\left(X_1 \quad X_2 \quad X_3 \quad X_4 \quad \cdots \quad X_{nvar}\right)\cdots$$

```
Sorting cost according to cost1
[f1, ind1] = sort (fout(:,1)); sorts objective function values
f2 = fout (ind1, 2);
[f2, ind2] = sort (fout(:,2));
f3 = fout(ind2, 3);
Chro = Chro (ind2,:); Sort the chro using induces
assign chrom on Pareto front cost = rank = 1
rank = rank + 1 and so on.
Replicate this chromosomes to get full number of population.
```

3.4. Parent Selection

During each successive generation, a proportion of the existing population is selected to breed a new generation.

Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection and rank based selections [18].

3.4.1. Roulette Wheel Selection

In roulette wheel selection, individuals are given a probability of being selected that is directly proportionate to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring.

Roulette Wheel's Selection Pseudo Code:

```
    for all members of population
    sum = fitness of individuals;
    end for
for all members of population
probability = sum of probabilities + (fitness/sum)
    sum of probabilities = probability
    end for
    loop until new population is full ;do this twice
    number = Random between 0 and 1
        for all members of population
    if number > probability but less than next probability then you have been selected
            end for; end
        create offspring
    end loop
```

3.4.2. Tournament-Based Selection

For K less than or equal to the number of population, extract K individuals from the population randomly make them play a “tournament”, where the probability for an individual to win is generally proportional to its fitness.

$$M = \text{ceil}\left(\left(\text{pop size} - \text{keep}\right)/2\right); \text{number of mating}$$

$$\text{Picks} = \text{ceil}\left(\text{keep} \cdot \text{rand}\left(K, M\right)\right);$$

Repeat M time. The parameter for tournament selection is the tournament size Tour.

Tour takes values ranging from 2 to N_{pop} (number of individuals in population). Selection intensity increases with Tournament size.

3.5. Reproduction

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and mutation.

For each new solution to be produced, a pair of “parent” solutions is selected for breeding from the pool selected previously.

By producing a “child” solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its “parents”. New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated. These processes ultimately result in the next generation population of chromosomes that is different from the initial generation.

Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

3.5.1. Crossover

The most common type is single point crossover. In single point crossover, you choose a locus at which you swap the remaining alleles from one parent to the other. This is complex and is best understood visually.

As you can see, the children take one section of the chromosome from each parent.

The point at which the chromosome is broken depends on the randomly selected crossover point.

This particular method is called single point crossover because only one crossover point exists. Sometimes only child 1 or child 2 is created, but oftentimes both offspring are created and put into the new population.

Crossover does not always occur, however. Sometimes, based on a set probability, no crossover occurs and the parents are copied directly to the new population.

1) Crossover (Real Valued Recombination)

Recombination produces new individuals in combining the information contained in the parents

The simplest methods choose one or more points in the chromosome to mark as the crossover points.

Some of Real valued recombination (crossover).

- Discrete recombination
- Intermediate recombination
- Line recombination

Real valued recombination: a method only applicable to real variables (and not binary variables).

The variable values of the offspring’s are chosen somewhere around and between the variable values of the parents as:

offspring = $\text{parent}_1 + \lambda(\text{parent}_2 - \text{parent}_1)$ *is convex* **combination** *of parents.*

where $\lambda \varepsilon \left[-d, 1+d\right]$.

In intermediate recombination $d = 0$,

for extended intermediate recombination $d > 0$.

A good choice is $d = 0.25$. (in some literature)

Randomly selecting crossover point:

α = roundup (rand × Nvar)

We’ll let

$$\text{parent}_1 = \left[X_{m1} \quad X_{m2} \quad X_{m3} \quad \cdots \quad X_{m\alpha} \quad \cdots \quad X_{mNvar}\right]$$
$$\text{parent}_2 = \left[X_{d1} \quad X_{d2} \quad X_{d3} \quad \cdots \quad X_{d\alpha} \quad \cdots \quad X_{dNvar}\right]$$

where: the *m* and *d* subscripts discriminate between the *mom* and the *dad* parent. Then the selected variables are combined to form new variables that will appear in the children:

$$X_{\text{new}1} = X_{m\alpha} - \lambda\left[X_{m\alpha} - X_{d\alpha}\right]$$

$$X_{\text{new}2} = X_{d\alpha} + \lambda\left[X_{m\alpha} - X_{d\alpha}\right]$$

where λ is also a random value between 0 and 1

$$\lambda = \text{rand}$$

The final step is:

$$\text{Offspring}_1 = \left[X_{m1} \quad X_{m2} \quad X_{m3} \quad \cdots \quad X_{\text{new}1} \quad \cdots \quad X_{mNvar}\right]$$

$$\text{Offspring}_2 = \left[X_{d1} \quad X_{d2} \quad X_{d3} \quad \cdots \quad X_{\text{new}2} \quad \cdots \quad X_{dNvar}\right]$$

3.5.2. Mutation

After selection and crossover, you now have a new population full of individuals. Some are directly copied, and others are produced by crossover [19]. In order to ensure that the individuals are not all exactly or the same, you allow for a small chance of mutation. You loop through all the alleles of all the individuals, and if that allele is selected for mutation, you can either change it by a small amount or replace it with a new value. The probability of mutation is usually small. Mutation is, however, vital to ensuring genetic diversity within the population. We force the routine to explore other areas of the cost surface by randomly introducing changes, or mutations, in some of the variables. We chose a mutation rate, [19].

$$\text{Nmut} = \text{ceil}\left(\text{popsize} \cdot Nvar \cdot \text{mutrate}\right);$$

Randomly choose rows and columns of the variables to be mutated.

$$mrow = \text{ceil}\left(\text{rand}\left(1, Nmut\right) \cdot \left(\text{popsize} - \text{elite}\right)\right) + \text{elite};$$

$$mcol = \text{ceil}\left(\text{rand}\left(1, Nmut\right) \cdot Nvar\right);$$

A mutated variable is replaced by a new random number.
For Example

$$mrow = \begin{bmatrix}4 & 5 & 7 & 7\end{bmatrix}$$

$$mcol = \begin{bmatrix}2 & 2 & 1 & 5\end{bmatrix}$$

The first random pair is (4, 2).
Thus the value in row 4 and column 2 of the population matrix is replaced.
By $\text{mean}\left(\left[X_{ub} - X_{lb}\right]\right) \cdot \text{rand}\left(1, Nmut\right) + \text{mean}\left(\left[X_{lb}\right]\right)$

3.6. Elitism

With crossover and mutation taking place, there is a high risk that the optimum solution could be lost as there is no guarantee that these operators will preserve the fittest string. To counteract this, elitist preservation is used. Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. In an elitist model, the best individual from a population is saved before any of these operations take place. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution [20].

In this paper chromosome on Pareto front is used as elite children and not participate in mutation operations.

3.7. Parameters of Genetic Algorithm

Crossover probability: how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better [19].

Mutation probability: the probability of mutation is normally low because a high mutation rate would destroy fit strings and degenerate the genetic algorithm into a random search. Mutation probability value in some literature is around 0.1% to 0.01% are common. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes.

Population size: how many chromosomes are in population (in one generation). If there are too few chromosomes, GA has few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations.

3.8. Termination Conditions

Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached. Thus generational process is repeated until a termination condition has been reached.

Common terminating conditions can be:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Manual inspection
- Any Combinations of the above

4. Test Functions

Three-objective test functions for the Simulation of the algorithm. the red surface is non dominated solutions while the blue region is feasible solutions which is domenated by points on red surface.

1) (Problem from Kalyanmoy Deb 2001)

$$\text{Min } f_1 = 1 + (x_1 - 1)^5$$

$$\text{Min } f_2 = x_2$$

$$\text{Min } f_3 = x_3$$

$$\text{s.t. } x_3^2 + \left(1 + (x_1 - 1)^5\right)^2 - 0.5 \geq 0;$$

$$x_3^2 + x_2^2 - 0.5 \geq 0;$$

$$ub = [2,1,1]; \ lb = [0,0,0];$$

2) (Adugna, 2014) [19].

$$\text{Min } f_1 = x_1;$$

$$\text{Min } f_2 = \frac{(x_2 + 2)}{x_1}$$

$$\text{Min } f_3 = x_1^2 + X_2;$$

$$\text{s.t. } ub = [4,3]; lb = [0.1,1];$$

3) Unconstrained linear minimization (Adugna, 2014) [19]

$$\text{Min } f_1 = x_1 - 3x_2;$$

$$\text{Min } f_2 = 3x_1 + x_2$$

$$\text{Min } f_3 = x_3 - 2x_2;$$

$$\text{s.t. } ub = [2,2,2]; \ lb = [1,1,0];$$

4) Comet problem (Problem from Deb 2001)

$$\text{Min } f_1 = (1+x_3)\cdot\left((x_1)^3\cdot(x_2)^2 - 10x_1 - 4x_2\right);$$

$$\text{Min } f_2 = (1+x_3)\cdot\left((x_2)^3\cdot(x_2)^2 - 10x_1 + 4x_2\right);$$

$$\text{Min } f_3 = 3\cdot(1+x_3)\cdot(x_1)^2;$$

$s.t\ \ ub = [3.5, 2, 1];\ \ lb = [1, -2, 0];$

5)

$$\text{Min } f_1 = x_1^2 + (x_2 - 1)^2$$

$$\text{Min } f_2 = x_1^2 + (x_2 + 1)^2 + 1$$

$$\text{Min } f_3 = (x_1 - 1)^2 + x_2^2 + 2$$

s.t. $ub = [4, 3];\ \ lb = [0, 1]$

6) (Problem from R. Viennet, C Fonteix, and I. Marc.1996) [21].

$$\text{Min } f_1 = 0.5\left(x_1^2 + x_2^2\right) + \sin\left(x_1^2 + x_2^2\right)$$

$$\text{Min } f_2 = \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15;$$

$$\text{Min } f_3 = 1/\left(x_1^2 + x_2^2 + 1\right) - 1.1\mathrm{e}^{\left(x_1^2 - x_2^2\right)};$$

s.t. $ub = [2, 2.5];\ \ lb = [-1.8, -1];$

5. Discussions and Conclusions

Many search techniques required auxiliary information in order to work properly. For example Gradient techniques need derivative in order to minimize/maximize a given objective where as genetic algorithms do not require all these auxiliary information. GA uses probabilistic transition rules to guide their search towards regions of search space with likely improvement. It is also better than other optimization algorithm when domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space and bad proposals do not affect the end solution (independent of initial feasible solution).

In general, multi-objective optimization requires more computational effort than single-objective optimization.

Genetic algorithms require several heuristic parameters and this process is not necessarily straightforward; it may require significant experience in selection. However, genetic multi-objective algorithms are relatively easy to use then other evolutionary algorithm.

In this paper algorithm is tested with linear, non-linear three objective functions and with constrained objectives functions after external penalty is carefully selected and the Simulation results from test functions show that the performance is quite satisfactory. The result on the approximation of the test function described in this paper is still preliminary. Further work should be carried out to check the conjectures on multiobjective optimization problems with a higher design space.

References

[1] Sawaragi, Y., Nakayama, H. and Tanino, T. (1985) Theory of Multiobjective Optimization. Academic Press, Waltham.

[2] Steuer, R.E. (1986) Multiple Criteria: Theory, Computation and Application. John Wiley & Sons, New York.

[3] Ehrgott, M. (2005) Multicriteria Optimization. Springer, New York.

[4] Eichfelder, G. (2008) Vector Optimization. Springer, Berlin Heidelberg.

[5] Schaffer, J.D., Ed. (1989) Advances in Genetic Programming. *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, Burlington.

[6] Coley, D.A. (1999) An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific Publishing Co. Pte. Ltd., River Edge.

[7] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. (2000) A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 849-858.

[8] Fonseca, C.M. and Fleming, P.J. (1995) An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation Journal*, **3**, 1-16.

[9] Knowles, J.D. and Corne, D.W. (1999) The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. *Proceedings of the* 1999 *Congress on Evolutionary Computation*, Washington DC, 06 Jul-09 Jul 1999.

[10] Bot, R.I., Grad, S.-M. and Wanka, G. (2009) Duality in Vector Optimization. Springer-Verlag, Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-02886-1

[11] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston.

[12] Deb, K. (2001) Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Hoboken.

[13] Sivanandam, S.N. and Deepa, S.N. (2008) Introduction to Genetic Algorithms. Springer-Verlag, Berlin Heidelberg.

[14] Semu, M. (2003) On Cone D.C Optimization and Conjugate Duality. *Chinese Annals of Mathematics*, **24**, 521-528. http://dx.doi.org/10.1142/S0252959903000529

[15] Goh, C.J. and Yang, X.Q. (2002) Duality in Optimization and Variational Inequalities. Taylor and Francis, New York. http://dx.doi.org/10.1201/9781420018868

[16] Holland, J.H. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor. (2nd Edition, MIT Press, 1992.)

[17] Haupt, R.L. (2004) Sue Ellen Haupt: Practical Genetic Algorithms. 2nd Edition, John Wiley & Sons, Inc., Hoboken.

[18] Koza, J.R. (1992) Genetic Programming. Massachusetts Institute of Technology, Cambridge.

[19] Fita, A. (2014) Multiobjective Programming with Continuous Genetic Algorithm. *International Journal of Scientific & Technology Research*, **3**, 135-149.

[20] Ikeda, K., Kita, H. and Kobayashi, S. (2001) Failure of Pareto-Based MOEAs: Does Nondominated Really Mean Near to Optimal. *Proceedings of the Congress on Evolutionary Computation*, **2**, 957-962.

[21] Viennet, R., Fonteix, C. and Marc, I. (1996) Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. *International Journal of Systems Science*, **27**, 255-260.

[22] http://www.matworks.com

Appendix A

%some of the concept of the m-file is taken from the MATWORKS web site www.matworks.com (genetic tool box) and modified by researcher for three objective programming [22].

```
%_____________________________________________________________
function f = ParetoGA3
% I parameters% variable Setup
clear all; warning off;
ub=[2,2.5]; lb=[-1.8,-1 ]; % variable limits
A=[lb;ub];% matri0x of the varriable bounds
nvar=length(A); % number of optimization variables Generations =20; % max number of iterations
selection=0.5; % fraction of population kept
Npop=2500;% populationsize
keep=selection*Npop; M=ceil((Npop-keep)/2); % number of matings
Pm=0.3;         % mutation rate
%_____________________________________________________________
%II Create the initial population
t=0; % generation counter initialized
phen=intpop(lb, ub,Npop,nvar);    % random population of    continuous values in matrix A
%_____________________________________________________________
%% III Evaluation of random population with objective function:
  fout=feval('parto3',phen); % Iterate through generations
while t<Generations
t=t+1;% increments generation counter
[f1,ind1]=sort(fout(:,1));%sorts objective function values
f2=fout(ind1,2); [f2,ind2]=sort(fout(:,2));
f3=fout(ind2,3); phen=phen(ind2,:); r=0; rank=1;
elt=0;
while r<Npop
for j=1:Npop
    if f3(j)<=min(f3(1:j))
r=r+1; elt(r)=j; value(r,1)=rank;end%if
if rank<=1
f11=f1(elt);f21=f2(elt);f31=f3(elt); elite=length(elt);
end%if
if r==Npop    break; end%if end%for
j=j+1; rank = rank+1;
end%while
phen=phen(elt,:);    % sorts chromosomes
value = value+1;
[value,ind]=sort(value);phen=phen(ind,:);
%_____________________________________________________________
%parent selection
Ntourn=Npop/4;
players=ceil(keep*rand(Ntourn,M));
[c,pt]=min(value(players));
for ib=1:M
mam(ib)=players(pt(ib),ib); end
players=ceil(keep*rand(Ntourn,M)); [c,pt]=min(value(players));
for ib=1:M
dad(ib)=players(pt(ib),ib);end
%_____________________________________________________________
%v. real valued Recombination
```

```
ii=1:2:keep; % index of mate #1
pt=floor(rand(1,M)*nvar); % crossover point
p=rand(1,M); % intermidate crossover
mix=phen(mam+Npop*pt)-phen(dad+Npop*pt);% mix from ma and pa
phen(keep+ii+Npop*pt)=phen(mam+Npop*pt)-p.*mix;% 1st offspring
phen(keep+ii+1+Npop*pt)=phen(dad+Npop*pt)+p.*mix;% 2nd offspring
if pt<nvar% crossover when last variable is selected
phen(keep+ii,:)=[phen(mam,1:pt) phen(dad,pt+1:nvar)];
phen(keep+ii+1,:)=[phen(dad,1:pt) phen(mam,pt+1:nvar)]; end % if
%____________________________________________________________
%vi.    Real valued Mutation
Nmut=ceil((Npop-elite)*nvar*Pm);% total number of mutations
mrow=ceil(rand(1,Nmut)*(Npop-elite))+elite;
mcol=ceil(rand(1,Nmut)*nvar);
mutindx=mrow+(mcol-1)*Npop;
phen(mutindx)= (ub(mcol)-lb(mcol)).*rand(1,Nmut)+lb(mcol);%to keep the randam number within ub and lb
%____________________________________________________________
% vii.The new offspring and mutated chromosomes are evaluated for cost
row=sort(rem(mutindx,Npop));
iq=1; rowmut(iq)=row(1);
for ic=2:Nmut
        if row(ic)>rowmut(iq)
iq=iq+1; rowmut(iq)=row(ic);
        if row(ic)>keep;break;end %if end %if end %for
if rowmut(1)==0;rowmut=rowmut(2:length(rowmut));end %if
fout(rowmut,:)=feval('parto3',phen(rowmut,:));
fout(keep+ii:Npop,:)=feval('parto3',phen(keep+ii:Npop,:));
fout(keep+ii+1:Npop,:)=feval('parto3',phen(keep+ii+1:Npop,:));
%____________________________________________________________
% Stopping criteria
if t>Generations
break; end %if
%____________________________________________________________
% Displays the output
phen(1:10,:)
[phen(elt,:)    fout]
figure(3);plot3(f2,f1,f3,'.',f21, f11,f31,'r*', 'LineWidth',2);xlabel('f2');ylabel('f1');zlabel('f3');grid on;
hold on;
comet3(f21, f11,f31);hold off;
% plot pareto fronterif3 index of objective function is two
title(['pareto fronter with Generation=',num2str(t)]);hold on;
%figure(2);plot(phen(ind,1),phen(ind,2),'b.');xlabel('x1');ylabel('x2');grid on;
title(['decision space with Generation=',num2str(t)]),pause(0.01);
end %t
format short g
disp(['Pareto front and objective value'])
disp(['         x1              x2                 f(x1)              f(x2)'])
disp(num2str(phen(elt,:)))
% generating matrices of initial population
function phen=intpop(lb,ub,Npop,nvar) %
for ii=1:Npop
phen(ii,:)=(ub-lb).*rand(1, nvar) + lb; end
```

Scientific Research

www.scirp.org

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.