Multiobjective Programming With Continuous Genetic Algorithm

Adugna Fita

Abstract: Nowadays, we want to have a good life, which may mean more wealth, more power, more respect and more time for our selves, together with a good health and a good second generation, etc. Indeed, all important political, economical and cultural events have involved multiple criteria in their evolution. Multiobjective optimization deals with the investigation of optimization problems that possess more than one objective function. Usually there is no single solution that optimizes all functions simultaneously, we have solution set that is called nondominated set and elements of this set are usually infinite. It is from this set decision is 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. This Paper will try to solve problems that the decision maker face in extraction of Pareto optimal solution with continuous variable genetic algorithm and taking objective function as fitness function without modification by considering box constraint and generating initial solution within box constraint and penalty function for constrained one. 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 Literature Survey.

EVERYDAY we encounter varies kinds of decision making problems as manager, resource planner, designers, administrative officers, mere individuals, and so on. In these problems, the final decision is usually made through several steps; the structural model, the impact model, and the evaluation model even though they sometimes not be perceived explicitly. Though the process, the objective of the problem and alternatives to perform it are specified. Hereafter, we shall use the notation Y for the objective and X for the set of alternatives, which is supposed to be a subset of an n-dimensional vector space. 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. Now we can formulate our decision making problems as: a Multiobjective optimization problem:

$$\text{Minimize} f(x) = (f_1(x), f_2(x), \dots, f_p(x))^T, \text{over}, x \in X. \text{b} \quad (1.1)$$

Where:

$$\mathrm{X} = \{\mathrm{x} \in \mathrm{R^n} : \mathrm{g_j(x)} \leq 0, \mathrm{j} = 1,2, \dots, \mathrm{m}, \mathrm{x} \geq 0 \quad (1.2)$$

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

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

$$g_j(x) \leq 0, j = 1,2, \dots, m, \quad (1.4)$$

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 $h_k(x) \leq 0 \; and \; -h_k(x) \leq 0$, and hence, it does not appear explicitly in this paper. (*R*. E. Steuer, *(1986)*, 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 contain the image of the feasible set $Y = f(X)$ is referred to as criterion space. 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. Here we assume a decision maker who is responsible for the final decision. [Y. Sawaragi, H. Nakayama and T. Tanino*, 1985*] The decision maker's value is usually represented by saying whether or not an alternative x is preferred to another alternative x' or equivalently whether or not $f(x)$ is preferred to (x') . In other words, the decision maker's value is represented by some binary relation over X or $f(X)$. Since such a binary relation representing the decision maker's preference usually become an order, it is called *a preference order* and it is supposed to be defined on the so called criteria space Y, which includes the set $f(X)$. Several kind of preference orders could be possible, sometimes the decision maker cannot judge whether or not $f(x)$ is preferred to $f(x')$. Such an order that admits incomparability for a pair of objects is called *partial order*, whereas the order requiring the comparability for every pair of objects is called *a weak order or total order*. In practice, we often observe a partial order for the decision maker's preference. Unfortunately,

- *Adugna Fita is currently lecturer and researcher at Adama Science and Technology University, Department of Mathematics, Adama, Ethiopia*
- *E-mail: fitaadu@yahoo.com*

however, an optimal solution in the sense of one that is more preferred with respect to the order, hence the notion of optimality does not necessarily exist for partial orders. Instead of strict optimality, we introduce in multiobjective optimization the notion of efficiency. A vector $f(x')$ is said to be efficient if there is no $x \in X$ such that $f(x)$ is preferred to $f(x')$ with respect to the preference order. The final decision is made among the set of efficient solutions. 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 [13] weighting parameter vector, the entries of which are the specific weights for the individual objective functions. Solutions to these problems constitute so-called properly efficient points with respect to our initial multiobjective optimization problem. The advantage of this approach is it can be solved efficiently by well-known algorithms, for example by interior-point methods. Using this algorithm a single solution is obtained at a time. To get all Pareto front we have to run the algorithm many times by applying different weigh. A *fundamental challenge lies in the selection of the 'right' weighting parameters* that lead to different optimal solutions. The other problem is assigning large weight to the best objective does not result in desired solution. Besides, stochastic methods and evolutionary approaches as well as different deterministic optimization techniques such as branch and bound algorithms for example are other established strategies for solving multiobjective optimization problems. Vector evaluated genetic algorithm (VEGA) Schaffer (1985) presents one of the first treatments of multi-objective genetic algorithms, although he only considers unconstrained problems. 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, [1], [4]. One sub-population is created by evaluating one objective function at a time rather than aggregating all of the functions. The selection process is composed of a series of computational loops, and during each loop, the fitness of each member of the population is evaluated using a single objective function. Then, certain members of the population are selected and passed on to the next generation, using the stochastic selection processes. This selection process is repeated for each objective function. 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). Such minima generally define the vertices of the Pareto optimal set. However, considering only one objective function at a time is comparable to setting all but one of the weights to zero. The vector of weights represents a search direction in the criterion space, and if only one component of the vector is non-zero, then only orthogonal search directions, parallel to the axes of the criterion space, are used (Murata et al. 1996). Such a process can be relatively ineffective. 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. The means of determining rank and assigning fitness values associated with rank may vary from method to method, but the general approach is common as described below. For a given population, the objective functions are evaluated at each point. All non-dominated points receive a rank of one. Determining whether a point is dominated or not (performing a non-dominated check) entails comparing the vector of objective function values at the point to the vector at all other points. Then, the points with a rank of one are temporarily removed from consideration and the points that are non-dominated relative to the remaining group are given a rank of two. This process is repeated until all points are ranked. Those points with the lowest rank have the highest fitness value. That is, fitness is determined such that it is inversely proportional to the rank. This may be done using any of several methods [7]; Srinivas and Deb 1995; Cheng and Li 1998; Narayanan and Azarm 1999). Belegundu et al. (1994) suggest discarding points with higher ranks and immigrating new randomly generated points. It is possible to have a Pareto optimal point in a particular generation that does not survive into the next generation. In response to this condition, Cheng and Li (1997) propose the use of a Pareto-set filter, which is described as follows. The algorithm stores two sets of solutions: a current population and a filter. The filter is called an approximate Pareto set, and it provides an approximation of the theoretical Pareto optimal set. With each generation, points with a rank of one are saved in the filter. When new points from subsequent generations are added to the filter, they are subjected to a non-dominated check within the filter, and the dominated points are discarded. Ishibuchi and Murata (1996), and Murata et al. (1996) use a similar 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. 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). Develop the tournament selection technique for the selection process, which proceeds as follows. 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. [2],[4]. A niche in genetic algorithms is a group of points that are close to each other, typically in the criterion space. Niche techniques (also called niche schemes or niche-formation methods) are methods for ensuring that a population does not converge to a niche, i.e., a limited number of Pareto

points. Thus, these techniques foster an even spread of points (in the criterion space). Genetic multi-objective algorithms tend to create a limited number of niches; they converge to or cluster around a limited set of Pareto points. 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 (Goldberg and Richardson 1987; Deb 1989; Srinivas and [3] 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,[4]. Narayana and Azarm (1999) present a method in which a limit is placed on the Euclidean distance in the criterion space between points (parents) that are selected for crossover. If the distance is too small, then the parents are not selected for crossover. In addition, the authors suggest that only non-dominated points (points with a rank of 1) be evaluated for constraint violation. Those points that violate constraints then receive a fitness penalty, which is imposed by reassigning their rank to be a large number (e.g., the population size). However, Pareto ranking is the most appropriate way to generate an entire Pareto front in a single run of a Genetic algorithm and its main advantage is that the approach is less susceptible to the shape or continuity. The primary questions when developing genetic algorithms for multi-objective problems are how to evaluate fitness, how to determine the potential solution points that should be passed on to the next generation, and how to incorporate the idea of Pareto optimality. The approaches that are described so far collectively address these issues. 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. 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 [16]. 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. Mathematical preliminaries

In this chapter we investigate optimization problems with feature more than one objective function. In the beginning fundamental concepts that are essential for multiobjective optimization are established. That is, cone, and ordering cones are introduced and key properties of efficient sets are examined in detail. Subsequently, the existence of efficient points and techniques to compute these are analyzed.

2.1 Convex set

Definition 2.1.1 *(algebraic sum of two sets)* [9]

1. *The algebraic sum of two sets is defined as* $S^1 + S^2 := \{z^1 + z^2 \mid z^1 \in S^1, z^2 \in S^2$ *In case* $S^1 = \{z^1\}$ *is a singleton we use the form* $z^1 + S^2$ *instead of* $\{z^1\} + S^2$.
2. *Let* $s, s^1, s^2 \subset \mathbb{R}^p$ *and* $\alpha \in \mathbb{R}$. *The multiplication of a scalar with a set is given by* $\alpha S := \{\alpha z \mid z \in S\}$, *in particular* $-S\ \{-z \mid z \in S\}$

Definition 2.1.2

i. *The point* $X \in \mathbb{R}^n$ *is said to be a convex combination of two points* $x^1, x^2 \in \mathbb{R}^n$ *if* $X = \alpha x^1 + (1 - \alpha)\, x^2$, *for some* $\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$.

ii. *The point* $X \in \mathbb{R}^n$ *is said to be a convex combination of* m *points* $x^1, x^2, \dots, x^m \in \mathbb{R}^n$ *if*

$$X = \sum_{i=1}^{m} \alpha_i x^i, for\ \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1\,.$$

Definition 2.1.3 *A set* $C \subset \mathbb{R}^n$ *is said to be convex if for any* $x^1, x^2 \in C$ *and every real number* $\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$, *the point* $\alpha x^1 + (1 - \alpha)\, x^2 \in C$

In other words, C is convex if the convex combination of every pair of points in C lies in C.
The intersection of all convex sets containing a given subset C of $\mathbb{R}^n$ is called the convex hull of C and denoted by $\mathrm{conv}(C)$.

Definition 2.1.4 (Cone) $C \subset \mathbb{R}^p$ is called a cone if $\alpha c \in C$ for all $c \in C$ *and for all* $\alpha \in \mathbb{R}, \alpha > 0$. A cone C is referred to as:

- Nontrivial or proper, if $C \neq \emptyset\ and\ C \neq \mathbb{R}^p$.
- Convex, if $\alpha d^1 + (1-\alpha) d^2 \in C$ for all $d^1\ and\ d^2 \in C$ for all $0 < \alpha < 1$
- Pointed, if for $d \in C, d \neq 0, -d \notin C, i\,e.\, C \cap -C \subseteq \{0\}$

Theorem 2.1.1 *A cone* C *is convex if and only if it is closed under addition. In other words,*

$\alpha c^1 + (1-\alpha) c^2 \in C \Leftrightarrow c^1 + c^2 \in C\ \forall\, c^1, c^2 \in C,$
$\forall \alpha \in [0, 1]$

Proof first, assume that the cone C is convex. Then we can conclude for $c^1,\ c^2 \in C$ and $\alpha = 1/2$ in combination with the cone property of C that $\left(\frac{1}{2}\right) c^1 + \left(1 - \frac{1}{2}\right) c^2 \in C$

$$2\left(\left(\frac{1}{2}\right) c^1 + \left(\frac{1}{2}\right) c^2\right) \in C$$

$$c^1 + c^2 \in C.$$

Secondly, suppose that the cone C is closed under addition. Exploiting the fact that C is a cone we deduce for $c^1, c^2 \in C$ and $\alpha \in [0, 1]$ that $\alpha c^1 \in C$ and $(1-\alpha)\, c^2 \in C$. Furthermore, since the cone is closed under addition we derive for these elements that

2.2 Preference Orders and Domination structures

A preference order represents the preference attitude of the decision maker in the objective space. It is a binary relation on a set $\mathrm{Y} = f(X) \subset \mathbb{R}^{\mathrm{p}}$ where f is a vector valued objective

function, and X is a feasible decision set. The basic binary relation $>$ means strict preference i.e $y > z$ for $y, z \in Y$ means objective value y is preferred to z.

Definition 2.2.1 (Binary relation) *Let S be any set. A binary relation R on S is a subset of $S \times S$. It is called [5]*

1. $reflexive, if\ (s,s) \in R$ *for all* $s \in S$,
2. $2.irreflexive\ if\ (s,s) \notin R$ *for all* $s \in S$,
3. $3.symmetric\ if\ (s^1,s^2) \in R \Rightarrow (s^2, s^1) \in R$ *for all* $s^1, s^2 \in S$,
4. $4.asymmetric\ if\ (s^1,s^2) \in R \Rightarrow (s^2, s^1) \notin R$ *for all* $s^1, s^2 \in S$,
5. $5.antisymmetric, if\ (s^1,s^2) \in R$ *and* $(s^2, s^1) \in R \Rightarrow s^1 = s^2$ *for all* $s^1, s^2 \in S$,
6. $6.transitive, if\ (s^1,s^2) \in R$ *and* $(z^2, z^3) \in R \Rightarrow (s^1, s^3) \in R$ *for all* $s^1, s^2, s^3 \in S$,
7. $7.negatively\ transitive\ if\ (s^1, s^2) \notin R$ *and* $(s^2, s^3) \notin R \Rightarrow (s^1, s^3) \notin R$ *for all* $s^1, s^2, s^3 \in S$,
8. $8.connected\ if\ (s^1,s^2) \in R$ *or* $(s^2, s^1) \in R$ *for all* $s^1, s^2 \in S$ *with* $s^1 \neq s^2$,
9. $9.$ strongly connected (or total) if $(s^1, s^2) \in R$ or $(s^2, s^1) \in R$ for all $s^1, s^2 \in S$.

Definition 2.2.2 *A binary relation R on a set S is a preorder (quasi-order) if it is reflexive and transitive. Instead of* $(s^1, s^2) \in R$ *we shall also write* $s^1 R\ s^2$. *In the case of* R *being a preorder the pair* (S, R) *is called a preordered set. In the context of (pre)orders yet another notation for the relation R is convenient. We shall write* $s^1 \leq s^2$ *as shorthand for* $(s^1, s^2) \in R$ *and* $s^1 \not\leq s^2$ *for* $(s^1, s^2) \notin R$ *and indiscriminately refer to the relation R or the relation* $\leq$. *This notation can be read as "preferred to".*[6],[2] Given any preorder $\leq$, two other relations are closely associated with $\leq$.

We define them as follows:
$s^1 < s^2 \Leftrightarrow s^1 \leq s^2$ and $s^2 \not\leq s^1$,
$s^1 \sim s^2 \Leftrightarrow s^1 \leq s^2$ and $s^2 \leq s^1$.

Actually, $<$ and $\sim$ can be seen as the strict preference and equivalence (or indifference) relation, respectively, associated with the preference defined by preorder $\leq$.

Definition 2.2.3 *A relation is named a* partial order *if it is reflexive, transitive and antisymmetric. If R solely satisfies the first two properties it is denoted as a* preorder *(or quasi-order).* Preference order (and more generally, binary relationship) on a set Y can be represented by a point-to-set map from Y in to Y. in fact, a binary relationship may be considered to be a subset of the product set YxY, and so it can be regarded as a graph of a point-to-set map from Y in to Y. That means we identify the preference order $\succ$ with the graph of set valued map P: [13],[15]

$$P(y) := \{y' \in Y \mid y \succ y'\}$$

($P(y)$ is the set of elements in Y less preferred to y) [C. J. Goh and X. Q. Yang, (2002)], Another approach for specifying preferences of the decision maker are the so called domination structures (ordering cone) where the preference order is represented by a set-valued map. Therefore for all $y \in \mathbb{R}^p$, we define the set of domination factors

$$C(y) := \{d \in \mathbb{R}^p \mid y \succ y + d\} \cup \{0\}$$

is defined where $y \succ y'$ means that the decision maker prefers y more than y'. A deviation of $d \in C(y)$ from y is hence less preferred than the original y. The most important and interesting special case of a domination structure is when $C(.)$ is a constant set-valued map, especially if $C(y)$ equals a pointed convex cone for all $y \in \mathbb{R}^p$ i.e. $C(.) = C$. A cone C is called pointed if $C \cap (-C) = \{0p\}$.Given an order relation R on $\mathbb{R}^p$, we can define a set

$$C_R := \{y^2 - y^1 : y^1 R\, y^2\}, (**)$$

Which we would like to interpret as the set of nonnegative elements of $\mathbb{R}^p$ according to R It is interesting to consider the definition (**) with $y^1 \in \mathbb{R}^p$, fixed, i.e., $C_R(y^1) = \{y^2 - y^1 : y^1 R\, y^2\}$. If R is an order relation, $y^1 + C_R(y^1)$ is the set of elements of $\mathbb{R}^p$, that y^1 is preferred to (or that are dominated by y^1). A natural question to ask is: Under what conditions is C_R (y) the same for all $y \in \mathbb{R}^p$? In order to answer that question, we need another assumption on order relation R.

Definitions 2.2.4 [5]

1. A binary relation R is said to be compatible with $d^1 = y^2 - y^1 \in C_R$addition if $(y^1+z,\ y^2+z) \in R$ for all $z \in \mathbb{R}^p$, and all $(y^1,\ y^2) \in R$.
2. A binary relation R is said to be compatible with scalar multiplication if $(\alpha s^1,\ \alpha s^2) \in R$ holds for all$(s^1,s^2) \in R$ and for all $\alpha \in \mathbb{R}$, and $\alpha > 0$

Theorem ***2.2.1*** *Let R be a relation which is compatible with scalar multiplication, then C_R is a cone.*

Proof Let $d \in C_R$ then $d = y^2 - y^1$ for some $y^1, y^2 \in \mathbb{R}^p$ with $(y^1, y^2) \in R$. Thus, $(\alpha y^1, \alpha y^2) \in R$ for all $\alpha > 0$, Hence $\alpha d = \alpha (y^2 - y^1) = \alpha y^2 - \alpha y^1 \in C_R$, for all $\alpha > 0$. □ The following result constitutes how certain properties of a binary relation R affects the characteristics of the cone C_R .

Lemma ***2.3.1*** If *R is compatible with addition and $d \in C_R$ then $0Rd$.*

Proof Let $d \in C_R$. Then there are $y^1, y^2 \in \mathbb{R}^p$ with $y^1 R\, y^2$ such that $d = y^2 - y^1$. Using $z = -y^1$, compatibility with addition implies $(y^1 + z)\, R\, (y^2 + z)$ or $0Rd$. The above Lemma indicted that if R is compatible with addition, the sets $C_R(y), y \in \mathbb{R}^p$, do not depend on y. In this report, we will be mainly concerned with this case.□

Theorem ***2.2.2*** *Let R be a binary relation on $\mathbb{R}^p$ which is compatible with scalar multiplication and addition. Then the following statements hold*.

1. $0 \in C_R$ if and only if R is reflexive.
2. C_R is pointed if and only if R is antisymmetric.
3. C_R is convex if and only if R is transitive.

Proof [5] Let R be reflexive and let $y \in \mathbb{R}^p$. Then yRy and $y - y = 0 \in C_R$. Let $0 \in C_R$. Then there is some $y \in \mathbb{R}$ with yRy. Now let $y' \in \mathbb{R}^p$. Then y' = y +z for some $z \in \mathbb{R}^p$. Since yRy and R is compatible with addition we get $y'Ry'$ 2. Let R be antisymmetric and let $d \in C_R$ such that $-d \in C_R$, too. Then there are $y^1, y^2 \in \mathbb{R}^p$ such that $y^1 R y^2$ *and* $d = y^2 - y^1$ as well as $y^3, y^4 \in \mathbb{R}^p$ such that $y^3 Ry^4$ and $-d = y^4 - y^3$. Thus, $y^2 - y^1 = y^3 - y^4$ and there must be y $\in \mathbb{R}^p$ such that $y^2 = y^3 + y$ and $y^1 = y^4 + y$. Therefore compatibility with addition implies $y^2 Ry^1$. Antisymmetry of R now yields $y^2 = y^1$ and therefore, $d = 0$, i.e. C_R is pointed. Let $y^1, y^2 \in \mathbb{R}^p$ with $y^1 Ry^2$ and $y^2 Ry^1$. Thus, $d = y^2 - y^1 \in C_R$ and $-d = y^1 - y^2 \in C_R$. If C_R is pointed we know that $\{d, -d\} \subset C$ implies $d = 0$ and therefore $y^1 = y^2$, i.e., R is antisymmetric. 3. Let R be transitive and let $d^1, d^2 \in C_R$. Since R is compatible with scalar multiplication, C_R is a cone and we only need to show $d^1 + d^2 \in C_R$. By Lemma *2.3.1* we have $0Rd^1$ and $0Rd^2$. Compatibility with addition implies $d^1 R(d^1 + d^2)$, transitivity yields $0R(d^1 + d^2)$, from which $d^1 + d^2 \in C_R$. Let C_R be convex and let $y^1, y^2, y^3 \in R^p$ be such that $y^1 Ry^2$ and $y^2 Ry^3$ .Then and $d^2 = y^3 - y^2 \in C_R$. Because C_R is *convex,* $d^1 + d^2 = y^3 - y^1 \in C_R$ By Lemma 2.3.1 we get $0R(y^3 - y^1)$ and by compatibility with addition $y^1 Ry^3$. □ Example The weak component wise order $\leq$ is compatible with addition and scalar multiplication. $C_{\leq} = \mathbb{R}^p_+$ contains 0, is pointed, and convex. We have defined cone C_R given a relation R. We can also use a cone to define an order relation. The concept of a constant set-valued map defining a domination structure has a direct connection to partial orderings. We recall the definition of partial orderings

Definition *2.2.2* [2],[8

i) A nonempty subset $R \subset \mathbb{R}^P \times \mathbb{R}^P$ is called a binary relation on $\mathbb{R}^P$. We write xRy for $(x, y) \in R$.

ii) A binary relation $\leq$ on $\mathbb{R}^P$ is called a partial ordering on $\mathbb{R}^P$ if for arbitrary $w, x, y, z \in \mathbb{R}^P$:

(1) $x \leq x$ (Reflexivity),

(2) $x \leq y, y \leq z \Rightarrow x \leq z$ (Transitivity),

(3) $x \leq y, w \leq z \Rightarrow x + w \leq y + z$ (Compatibility with addition),

(4) $x \leq y, \alpha \in \mathbb{R}_+ \Rightarrow \alpha x \leq \alpha y$ (Compatibility with the scalar multiplication).

iii) A partial ordering $\leq$ on $\mathbb{R}^P$ is called antisymmetric if for arbitrary $x, y \in \mathbb{R}^P$ $x \leq y, y \leq x \Rightarrow x = y$.

A linear space $\mathbb{R}^{\mathrm{P}}$equipped with a partial ordering is called a partially ordered linear space. An example for a partial ordering on $\mathbb{R}^{\mathrm{P}}$ is the natural (or component wise) ordering $\leq_{\mathrm{p}}$ defined by

$$\leq_{\mathrm{p}} := \{(\mathrm{x}, \mathrm{y}) \in \mathbb{R}^{\mathrm{P}} \times \mathbb{R}^{\mathrm{P}} \mid \mathrm{x}_{\mathrm{i}} \leq \mathrm{y}_{\mathrm{i}} \text{ for all } \mathrm{i} = 1, \ldots, \mathrm{p}\}.$$

Partial orderings can be characterized by convex cones. Any partial ordering $\leq$ in $\mathbb{R}^{\mathrm{P}}$ defines a convex cone by [3] as $\mathrm{C} := \{\mathrm{x} \in \mathbb{R}^{\mathrm{P}} \mid 0\mathrm{p} \leq \mathrm{x}\}$ and any convex cone, then also called ordering cone, defines a partial ordering on $\mathbb{R}^{\mathrm{P}}$ by

$$\leq_{\mathrm{C}} := \{(\mathrm{x}, \mathrm{y}) \in \mathbb{R}^{\mathrm{P}} \times \mathbb{R}^{\mathrm{P}} \mid \mathrm{y} - \mathrm{x} \in \mathrm{C}\}.$$

For example the ordering cone representing the natural ordering in $\mathbb{R}^{\mathrm{P}}$ is the positive orthant $\mathbb{R}^{\mathrm{p}}_+$ A partial ordering $\leq_{\mathrm{C}}$ is antisymmetric if and only if C is pointed. Thus, preference orders which are partial orderings correspond to domination structures with a constant set-valued map being equal to a convex cone. The point-to- set map $C: Y \to \mathbb{R}^p$ reprsents preference order and we call C domination structure (ordering cone)

3. Properties And Existence Of The Efficient Solution

3.1 Efficient Solutions

The concept of optimal solutions to multiobjective optimization is not trivial. It is closely related to the preference attitudes of the decision makers. The most fundamental solution concept is that of efficient solution (non dominated, or noninferior) solution with respect to the domination structure of the decision maker. [5] Let $Y = f(X)$ be the image of the feasible set X and $y = f(x)$ for $x \in X$. Hence, a multiobjective optimization problem is given by

Minimize $f(x) = (f_1(x), \ldots, f_p(x))$ (3.1)

subject to $x \in X \subset \mathbb{R}^n$.

A domination structure representing a preference attitude of the decision maker is supposed to be given as a point-to-set map C from Y to $\mathbb{R}^n$ .Hence, the optimization problem (3.1) can be restated as: $''Min'' \; y = f(x)$ (3.2)

$s.t. y \in Y = f(X)$.

Definition 3.1.1 *(Efficiency via order relations) An element $y^* \in Y$ is denoted as efficient, if there exists no other element $y \in Y$, that differs from y^* and is less than it with respect to the employed ordering relation $\leq_C$. That is,* $\nexists \mathrm{y} \in \mathrm{Y} : \mathrm{y} \neq \mathrm{y}^*, \mathrm{y} \leq_{\mathrm{C}} \mathrm{y}^*$

Definition 3.1.2 [*R. I. Bot, S-M. Grad, G. Wanka, (2009)], (Eciency via ordering cones) Let Y be a set and C a related ordering cone. Then $y^* \in Y$ is called efficient, if there exists no $y \in Y$ such that $y^* - y \in C \backslash \{0\}$.Moreover, all efficient elements for problem (P) are combined into the efficient sen $E(Y, C) := \{y \in Y \mid Y \cap (y - C) = \{y\}\}$. In addition, if Y is the image of the feasible set X, $x^* \in X$ is labelled as Pareto optimal if $y^* \in Y$, satisfying $y^* = f(x^*)$ is efficient.*

Definition 3.1.3 *A feasible solution $x' \in X$ is called Pareto optimal, if there is no other $x \in X$ such that $f(x) \leq f(x')$. If x' is Pareto optimal, $f(x')$ is called nondominated point. If $x^1, x^2 \in X$ and $f(x^1) \leq f(x^2)$ we say x^1 dominates x^2 and $f(x^1)$ dominates $f(x^2)$.* A feasible vector $x' \in X$ is called efficient or Pareto optimal $(C = \mathbb{R}^p_+)$, if there is no other decision vector $x \in X$ such that $f_i(\mathrm{x}) \leq f_i(x')$ for all $i = 1, 2, \ldots, p$, and $f_i(x) \leq f_i(x')$ for at least one objective function. In this case, $(f(x') - \mathbb{R}^p_+) \cap (Y) = \{f(x')\}$ or equivalently

$$(Y - (f(x^{'})) \cap (-\mathbb{R}_{+}^{p}) = \{0\}. x^{'} \in X$$

is called weak Pareto optimal solution to the problem if there does not exist another decision vector $x \in X$ such that $f(x) < f(x^{'})$. This means,

$(Y - (f(x^{'})) \cap (-intC) = \{0\}$. (3.2)

$(Y - (f(x^{'})) \cap (-C \setminus \{0\}) = \{0\}$. (3.3)

$(Y - (f(x^{'})) \cap (-int\mathbb{R}_{+}^{p}) = \emptyset$ (3.4)

(If there is no other decision vector $x \in X$ such that

$f_i(x) < f_i(x^{'})$ for all $i = 1,2,\ldots,p$.

Proposition3.1.1 *Let Y and Z be two sets in $\mathbb{R}^p$, and let C be constant ordering cone on $\mathbb{R}^p$ then $E(Y+Z,C) \subset E(Y,C) + E(Z,C)$*

Proof Let $y^* \in E(Y+Z,C)$, then $y^* = y + z$ for some $y \in Y, z \in Z$.We want to show $y \in E(Y,C)$ and $z \in E(Z,C)$. Suppose not, then there exist $y^{'} \in Y$ and $0 \neq d \in C$ such that $y = y^{'} + d$. Then $y^* = y^{'} + z + d$ and $y^{'} + z \in Y + Z$ which contradict the supposition $y^* \in E(Y+Z,C)$. Similarly we can show for Z. □

Theorem 3.1.1 *Let C be a pointed convex cone, then*

$$E(Y,C) = E(Y + C, C)$$

`

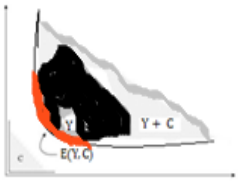

Figure1.

Proof The result is trivial if $Y = \emptyset$, because $Y + C = \emptyset$ and the nondominated subsets of both are empty, too. So let $Y \neq \emptyset$. First, assume $y \in E(Y + C, C)$, but $y \notin E(Y,C)$. There are two possibilities. If $y \notin Y$ there is $y^{'} \in Y$ and $0 \neq d \in C$ such that $y = y^{'} + d$. Since, $y^{'} = y^{'} + 0 \in Y + C$,we get $y \notin E(Y + C, C)$ wich is a contradiction. If $y \in Y$ there $is\ y^{'} \in Y$ such that $y^{'} \leq y$. Let $d = y - y^{'} \in C$. Therefore y = $y^{'}$+ d and $y \notin E(Y + C, C)$, again contradicting the assumption. Hence in either cases $y \in E(Y,C)$ Second, assume $y \in E(Y,C)$ but $y \notin E(Z + C, C)$. Then there is some $y^{'} \in Y + C$ with $y - y^{'} = d^{'} \in C$. That is $y^{'} = y^{''} + d^{''}$ with $y^{''} \in Y, d^{''} \in C$ and therefore $y = y^{'} + d^{'} = y^{''} + (d^{'} + d^{''}) = y^{''} + d$ with $d = d^{'} + d^{''} \in C$. This implies $y \notin E(Y,C)$, contradicting the assumption. Hence, $y \in E(Y + C, C)$. □

Theorem 3.1.2 *Let C be a nonempty ordering cone with $C \neq \{0\}$.Then $E(Y,C) \subseteq bd(Y)$.*

Proof Let ($C = \mathbb{R}_{+}^{p}$), and $y \in E(Y,C)$ and suppose $y \notin bd(Y)$. Therefore $y \in intY$ and there exists an ε-neighborhood $B(y,\varepsilon)$ of y (with $B(y,\varepsilon) := y + B(0,\varepsilon) \subset Y$, $B(0,\varepsilon)$ is an open ball with radius ε centered at the origin). Let $d \neq 0, d \in C$. Then we can choose some $\alpha \in \mathbb{R}, 0 < \alpha < \varepsilon$ such that $\alpha d \in B(0,\varepsilon)$. Now, $y - \alpha d \in Y$ with $\alpha d \in C \setminus \{0\}, i.e. y \notin E(Y,C)$. □

Theorem 3.1.3 *If Y is a closed and convex set and the section $(y - C) \cap Y$ is compact for all $y \in Y$, then $E(Y,C)$ is connected.* ***Proof [5]***

3.2 Existence of efficient solution

Definition 3.2.1 *Let $(S, \leq_C)$ be a preordered set, i.e. $\leq_C$ is reflexive and transitive. $(S, \leq_C)$ is inductively ordered, if every totally ordered subset of $(S, \leq_C)$ has a lower bound. A totally ordered subset of $(S, \leq_C)$ is also called a chain.*

Definition 3.2.2 *A set $Y \subset \mathbb{R}^p$ is called C-semicompact if every open cover of Y of the form $\{(y^i - clC)^c : y^i \in Y, i \in I\}$ has a finite subcover. For some indexed set IThis means that whenever $Y \subset \cup_{i \in I}(y^i - clC)^c$ there exist $m \in N$ and $\{i_1,\ldots,i_m\} \subset I$ such that $Y \subset \cup_{k=1}^{m}(y^{i_k} - clC)^c$ here $(y^i - clC)^c$ denotes the complement $C \setminus (y^i - clC)$ of $(y^i - clC)$. Note that these sets are always open.*

Theorem 3.2.1 *If C is an acute convex cone and Y is nonempty C-semicompact set in $\mathbb{R}^p$, then $E(Y,C) \neq \emptyset$.*

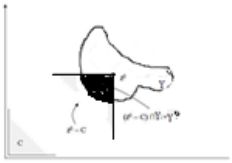

Figure. 2.

Proof [5] As $C \subset clC \Rightarrow E(Y, clC) \subset E(Y,C)$ It is enough to show the case in which C is a pointed closed convex cone. In this case, C define a partial order $\leq_c$ on Y as $y^1 \leq_c y^2$ iff $y -^2 y^1 \in C$. An element in $E(Y,C)$ is a minimal element with respect to $\leq_c$. therefore we can show that Y is inductively ordered and applying Zorn's lemma to establish the existence of minimal element. Now, suppose the contrary that Y is not inductively ordered, then there exist a totally ordered set $\bar{Y} = \{y^Y : Y \in \tau\}$ in Y which has no lower bound in Y. Thus $\cap_{Y \in \tau}[y^Y - C \cap Y] = \emptyset$.Otherwise any elements of this intersection is a lower bound of $\bar{Y}$in Y. Now it follows that for any $y \in Y$ there exist $y^Y \in \bar{Y}$ such that $y^Y \notin \bar{Y} - C$. since $y^Y - C$ is closed, the family of $\{(y^Y - C)^c : Y \in \tau\}$ forms an open cover of Y. Moreover, $y^Y - C \subset y^{Y^{'}} - C$ iff $y^Y \leq_c y^{Y^{'}}$, and so they are totally ordered by inclusion. Since Y is C-semicompact, the cover admits finite subcover, and hence there exist a single $y^Y \in \bar{Y}$ such that $Y \subset (y^{\bar{Y}} - C)^c$. However, this contradicts the fact that $y^Y \in Y$. Therefore Y is inductively ordered by $\leq_c$ and $E(Y,C) \neq \emptyset$ by Zorn's lemma. □ 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. 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 (search space). In addition to this continuity and differentiability of objective functions does not require.

4. Continuous Variable Genetic Algorithm.

4.1. 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 (1975), [12]. 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. Genetic algorithms are capable of giving rose to efficient and effective search in the problem domain. Hence the application is now widely spread in business, sciences and engineering. These algorithms are computationally less complex but more powerful in their search for improvement. These features have motivated the researchers to study different approaches of genetic algorithm.

4.2. Genetic Algorithm

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. Genetic algorithms is a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. [1] Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. 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.

4.2.1 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). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found. 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:

$$Pop = rand\ (Npop, Nvar)$$

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, [12]:

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

Where X_{lb}: lowest number in the variable range

X_{ub}: highest number in the variable range

X_{norm}: normalized value of variable.

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

for i=1:Npop

pop(i,:)=(X_{ub}−X_{lb}).*rand(1, nvar) + X_{lb}; end for;

This generate real valued population $N_{pop} \times N_{var}$ population matrix. Generally data encoding and generation of initial solution is problem dependent, possible individual’s encoding can be:

- Bit strings
- Real numbers
- Permutations of element; ... any data structure...

4.2.2 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.

Cost = f(chromosome) that means
$Cost_1 = f(X_1 X_2 X_3 X_4 \ldots. X_{nvar})$
$Cost_2 = f(X_1 X_2 X_3 X_4 \ldots. X_{nvar})$
$Cost_3 = f(X_1 X_2 X_3 X_4 \ldots. X_{nvar})$…
Sorting cost according to $cost_1$
[$cost_1$, ind] = sort(cost); % min cost in element 1
Chro = Chro (ind,:); % sort chromosome
cost1 = f1 (pop) ;
cost2 = f2 (pop);…
[cost, ind] = sort {cost1}
reorder based on sorted *cost1*
*cost2 =cost2 (*ind1*) ….*
pop =pop(ind1)
rank =1
Reorder based on sorted cost1 assign chromosome on Pareto front cost = rank remove all chromosome assigned the value of rank from the pop.
rank = rank +1

4.2.3 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.

4.2.3.1 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
```

4.2.3.2. 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=ceil ((pop size-keep)/2); number of mating

*Picks = ceil(keep*rand(K, M));*

Repeat M time. The parameter for tournament selection is the tournament size Tour. Tour takes values ranging from 2 - Npop (number of individuals in population). Selection intensity increases with Tournament size.

4.2.4 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.

4.2.5 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.

4.2.6 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 = parent 1 + λ(parent 2 − parent 1)
For example consider the two parents to be:
$parent_1 = [X_{m1}\, X_{m2}\, X_{m3}\, X_{m4} \ldots\, X_{mNvar}]$
$parent_2 = [X_{d1}\, X_{d2}\, X_{d3}\, X_{d4} \ldots\, X_{dNvar}]$
Crossover points are randomly selected, and then the variables in between are exchanged:
$Offspring_1 = [X_{m1}\, X_{m2}\, X_{d3}\, X_{m4} \ldots \downarrow X_{m7} \ldots\, X_{mNvar}]$
$Offspring_2 = [X_{d1}\, X_{d2} \uparrow X_{m3}\, X_{d4} \ldots \uparrow X_{d7} \ldots\, X_{dNvar}]$

The extreme case is selecting N_{var} points and randomly choosing which of the two parents will contribute its variable at each position. Thus one goes down the line of the chromosomes and, at each variable, randomly chooses whether or not to swap the information between the two parents.

This method is called uniform crossover:

$Offspring_1 = [X_{d1}\, X_{m2}\, X_{m3} \ldots\, X_{d6} \ldots\, X_{mNvar}]$
$Offspring_2 = [X_{m1}\, X_{d2}\, X_{d3} \ldots\, X_{m6} \ldots\, X_{dNvar}]$

The problem with these point crossover methods is that no new information is introduced: Each continuous value that was randomly initiated in the initial population is propagated to the next generation, only in different combinations and this strategy work fine for binary representations. There is now a continuum of values, and in this continuum we are merely interchanging two data points. The blending methods remedy this problem by finding ways to combine variable values from the two parents into new variable values in the offspring. A single offspring variable value, X_{new}, comes from a combination of the two corresponding offspring variable values:

$$X_{new} = \lambda X_{mn} + (1-\lambda) X_{dn}$$

Where:
λ: random number on the interval $[-a, a+1]$ *if* $a = 0$ *then* $\lambda \in (0,1)$
X_{mn}: *nth variable in the mother chromosome*
X_{dn}: *nth variable in the father chromosome*

The same variable of the second offspring is merely the complement of the first. If λ =1, the X_{mn} propagates in its entirety and X_{dn} dies. In contrast, if λ =0, then X_{dn} propagates in its entirety and X_{mn} dies. When λ =0.5, the result is an average of the variables of the two parents. Choosing which variables to blend is the next issue. Sometimes, this linear combination process is done for all variables to the right or to the left of some crossover point. Any number of points can be chosen to blend, up to N_{var} values where all variables are linear combination of those of the two parents. The variables can be blended by using the same λ for each variable or by choosing different λ's for each variable. However, they do not allow introduction of values beyond the extremes already represented in the population. Top do this requires an extrapolating method.

The simplest of these methods is linear crossover. In this case three offspring are generated from the two parents by:

$$X_{new1} = 0.5X_{mn} + 0.5\, X_{dn}$$

$$X_{new2} = 1.5X_{mn} - 0.5X_{dn}$$

Any variable outside the bounds is discarded in favor of the other two. Then the best two offspring are chosen to propagate. Of course, the factor 0.5 is not the only one that can be used in such a method. Heuristic crossover is a variation where some random number, λ, is chosen on the interval [0,1] and the variables of the offspring are defined by:

$$X_{new} = \lambda(X_{mn} - X_{dn}) + X_{mn}$$

Variations on this method include choosing any number of variables to modify and generating different λ for each variable. This method also allows generation of offspring outside of the values of the two parent variables. If this happens, the offspring is discarded and the algorithm tries another λ. The blend crossover method begins by choosing some parameter α that determines the distance outside the bounds of the two parent variables that the offspring variable may lie. This method allows new values outside of the range of the parents without letting the algorithm stray too far. The method used for us is a combination of an extrapolation method with a crossover method. We want to find a way to closely mimic the advantages of the binary GA mating scheme. It begins by randomly selecting a variable in the first pair of parents to be the crossover point:

*α = roundup(random*Nvar)*
We'll let
$parent1 = [X_{m1}\, X_{m2}\, X_{m3} \ldots\, X_{m\alpha} \ldots\, X_{mNvar}]$
$Parent\ 2 = [X_{d1}\, X_{d2}\, X_{d3} \ldots\, X_{d\alpha} \ldots X_{dNvar}]$

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:

$$X_{new1} = X_{m\alpha} - \lambda[X_{m\alpha} - X_{d\alpha}]$$
$$X_{new2} = X_{d\alpha} + \lambda[X_{m\alpha} - X_{d\alpha}]$$

Where λ is also a random value between 0 and 1, the final step is to complete the crossover with the rest of the chromosome as before:

$Offspring_1 = [X_{m1}\, X_{m2}\, X_{m3} \ldots\, X_{new1} \ldots\, X_{mNvar}]$
$Offspring_2 = [X_{d1}\, X_{d2}\, X_{d3} \ldots\, X_{new2} \ldots\, X_{dNvar}]$

If the first variable of the chromosomes is selected, then only the variables to the right to the selected variable are swapped. If the last variable of the chromosomes is selected, then only the variables to the left of the selected variable are swapped. This method does not allow offspring variables outside the bounds set by the parent unless λ >1.

Eg. Intermediate recombination
parent 1 : 12 25 5
parent 2 : 123 4 34

lambda are:
sample 1: 0.5 1.1 -0.1
sample 2: 0.1 0.8 0.5

The new individuals are calculated

offspring = parent 1 + λ (parent 2 − parent 1)
offspring 1: 67.5 1.9 2.1
offspring 2: 23.1 8.2 19.5

4.2.7 Mutation
After selection and crossover, you now have a new population full of individuals. Some are directly copied, and others are produced by crossover. 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,

*Nmut = ceil(popsize*Nvar*mutrate); Total number of mutations.*

Next random numbers are chosen to select the row and the columns of the variables to be mutated.

- mrow $= ceil(rand(1, Nmut) * (popsize - elite)) + elite;$
- $mcol = ceil(rand(1, Nmut) * Nvar);$

A mutated variable is replaced by a new random variable. The following pairs were randomly selected: for Example

mrow = [7 3 2 11]
mcol = [2 2 1 3]

The first random pair is (7,2). Thus the value in row 9 and column 2 of the population matrix is replaced with random number from the feasible region. To keep the solution in feasible region after mutation apply:

$$\text{Pop} = (\text{ub}(\text{mcol}) - \text{lb}(\text{mcol})).* \text{rand}(1, \text{Nmut}) + \text{lb}(\text{mcol});$$

3.2.8 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. After the new population is formed and evaluated, it is examined to see if this best structure has been preserved. If not, the saved copy is reinserted back into the population and the rest of the population is constructed in ways described above. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution [10]. In this paper chromosome on Pareto front is used as elite children and not participate in mutation operations.

4.2.8 Parameters Of Genetic Algorithm
There are two basic parameters of GA - crossover probability and mutation probability. 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 (but this does not mean that the new generation is the same!). Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old populations survive to next generation. 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, these values represent the probability that a certain string will be selected for mutation i.e. for a probability of 0.1%; one string in one thousand will be selected for mutation. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. 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. Mutation should not occur very often, because then GA will in fact change to random search. If the mutation rate is too high, the GA searching becomes a random search, and it becomes difficult to quickly converge to the global optimum. There are also some other parameters of GA. One another particularly important parameter is population size. 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.

4.2.9 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
- Allocated budget (computation time) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results

- Manual inspection
- Any Combinations of the above.

Simple GA flows chart

- Initialize P(t)
- Evaluate P(t)
- cost1 = f1 (P(t)) ;cost2 = f2 (P(t))
- [cost, indx1] = sort {cost1}
- reorder based on sorted cost1;
- cost2 =cost2 (indx1) ;
- P(t) =P((ind1)
- rank =1
- assign chrom .on Pareto front
- cost = rank
- remove all chrom . assigned the value of rank from the pop
- rank = rank +1
- Selection
- Apply crossover on P(t) to yield C(t)
- Apply mutation on C(t) to yield D(t)
- Select P(t+1)
- from P(t) and D(t) based on the fitness
- Stopping criteria
- *Stop*

4.2.10 Test functions

1. A two objective linear programming problem

$$Min\ f(1) = 3 * x(1) + x(2;$$
$$Min\ f(2) = -x(:,1) - 2 * x(2);$$
$$s.t. 3 * x(1) - x(2) \leq 6$$
$$x(2) \leq 3; x(1), x(2) \geq 0$$

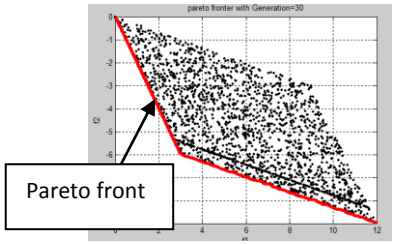

Figure 1a. Simulation Results

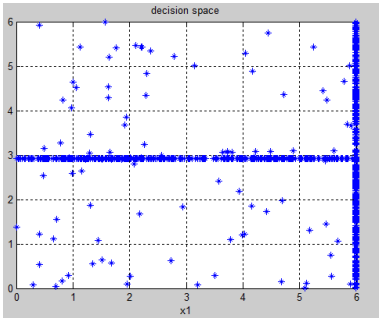

Figure 1b. Simulation Results

2. bi-objective nonlinear problem (Rendon)
Minimize $f1 = 1/((x1-3)^2 + (x2-3)^2 + 1)$;
Minimize $f2 = (x1-3)^2 + 3(x2-3)^2 + 1$;
s.t $[0 \leq x1, x2 \leq 6]$

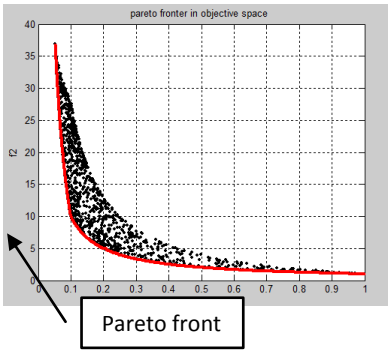

Figure 2a. Simulation Results

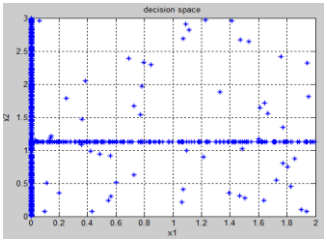

Figure 2b. Simulation Results

3. Two-objective warehouse location problem that minimize the total distance to five location so that the distance from fire station at (20,20) is less than 10 unit.(using Euclidian distance)

[(0,45);(20,45);(70,30);(80,5);(80,15);(20,20)]

%f(:,1)=sqrt((0-x(:,1)).^2+(45-x(:,2)).^2)+sqrt((5-x(:,1)).^2+(10-x(:,2)).^2)...
+sqrt((20-x(:,1)).^2+(45-x(:,2)).^2)+sqrt((70-x(:,1)).^2+(30-x(:,2)).^2)...
+sqrt((80-x(:,1)).^2+(5-x(:,2)).^2)+sqrt((80-x(:,1)).^2+(15-x(:,2)).^2);
f(:,2)= abs(sqrt((20-x(:,1)).^2+(20-x(:,2)).^2)-10);

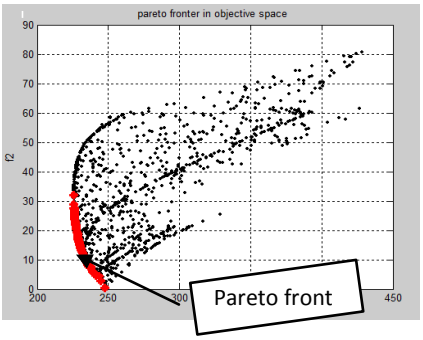

Figure 3a. Simulation Results

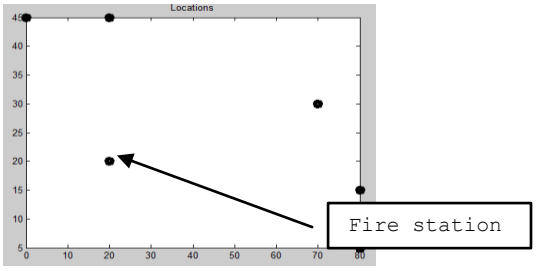

Figure 3b. locations

5 A two-objective with two variable minimization problem introduced by Deb (2001) is chosen to illustrate the function of SPEA2.

Minimize f1(x) = x1;
Minimize f2(x) =(x2 + 1)/x1
Subject to 0.1≤x1 ≤ 1; 0 ≤x2 ≤5:

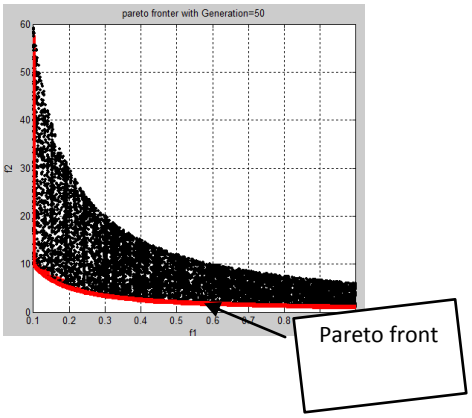

Figure 4a. Simulation Results

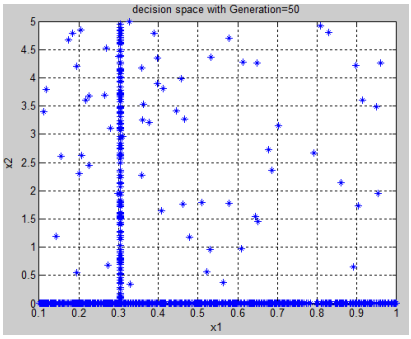

Figure 4b. Simulation Results

5. Constraint Handling a two-objective with two variable minimization problem introduced by Deb
(2001) is chosen to illustrate the function of SPEA2
Minimize f1(x) = x1;
Minimize f2(x) =(x2 + 1)/x1
Subject to $-x2 - 9x1 \leq -6$;
$x2 - 9x1 \leq -1$
0.1≤x1 ≤ 1; 0 ≤x2 ≤5 (Using external appropriate penalty)

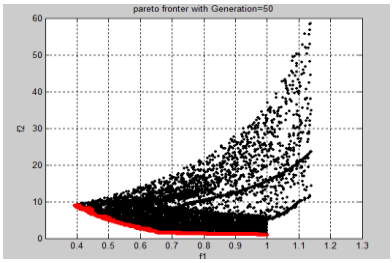

Figure 5a. Simulation Results

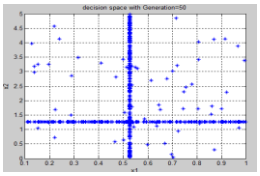

Figure 5b. Simulation Results

6. Three-Objective test problems
Min f1 = X1;
Min f2 = (X2 +2) /X1;
Min f3 = X1 ^{2} + X2;
ub=[4,3]; lb=[0.1,1];

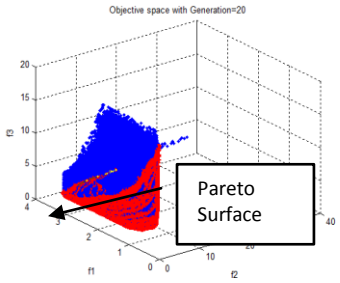

Figure 6. Simulation Results

7. Unconstrained linear minimization
$Min\ f1 = X1 - 3X2;$
$Min\ f2 = 3X1 + X2$
$Min\ f3 = X3 - 2X2;$
$ub = [2,2,2];\ lb = [1,1,0];$

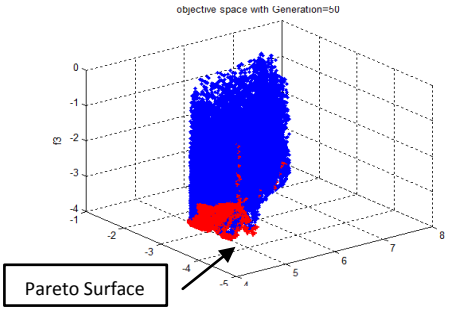

Figure 7. Simulation Results

8. Comet problem (Deb 2001)
Min F1 =(1+X$_3$)(X$_1$)^3 *(X$_2$)^2-10X$_1$-4X$_2$);*

Min f2 =(1+X_3)(X_1)^3 *(X_2)^2-10X_1+4X_2);*
Min f3 =3(1+X_3)*(X_1)^2;*
s.t ub=[3.5,2,1]; lb=[1,-2,0];

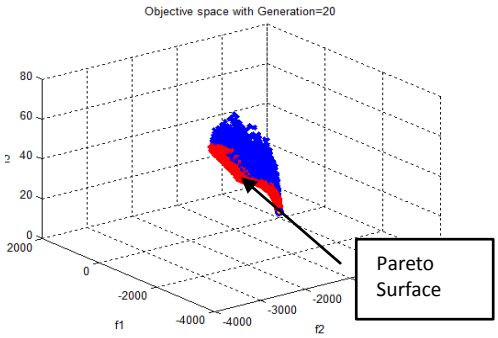

Figure8. Simulation Results

4.2.10 Discussion of using genetic algorithm

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 and other whereas genetic algorithms do not require all these auxiliary information. GA is blind to perform an effective search for better and better structures. This characteristic makes GA a more suitable method than many search schemes. GA uses probabilistic transition rules to guide their search towards regions of search space with likely improvement. Because of these important characteristics possessed by the GA it is more robust than other commonly used techniques when the search space is large, complex or loosely defined problems. 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.
- When no mathematical analysis is available.
- When traditional search methods fail.
- It can quickly scan a vast solution set.
- Bad proposals do not affect the end solution (independent of initial feasible solution)

4. Concluding remarks

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. Classical methods of preferences require the user to specify preferences only in terms of objective functions. Alternatively, genetic algorithm methods of preferences allow the user to view potential solutions in the criterion space or in the design space, and to select an acceptable solution that helps the decision maker in tradeoff analysis. In this paper algorithm is tested with linear, non-linear and with constrained objectives functions after external penalty is carefully selected and the performance is very good compared to classical method and some evolutionary algorithms. It is also simplifies the way fitness function is represented in other evolutionary algorithm.

5. 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. The algorithm can be work for function with more than two objectives with simple modification.
%_____________________________________________
function f = ParetoGA %for two objective
% I parameters Setup
warning off;
% ub = upper bound ;lb = lower bound of variables; % variable limits
A=[lb;ub];% matrix of the varriable bounds
nvar=length(A); % number of optimization variables
Generations =50; % max number of iterations
selection=0.5; % fraction of population kept
Npop=3000;% populationsize; keep=selection*Npop;
M=ceil((Npop-keep)/2); % number of matings
Pm=0.2; % 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('parto',phen)
% Iterate through generations
while t<Generations
t=t+1;% increments generation counter
[f1,ind]=sort(fout(:,1));%sorts objective function values
phen=phen(ind,:); % sorts chrom ; f2=fout(ind,2);
%f3=fout(ind,3);
r=0; rank=1; elt=0;
while r<Npop
for j=1:Npop
 if f2(j)<=min(f2(1:j))
r=r+1; elt(r)=j; value(r,1)=rank;end%if
if rank==1
f11=f1(elt);f21=f2(elt);elite=length(elt);end%if
if r==Npop break; end%if; end%for
rank = rank+1;;end%while; phen=phen(elt,:); % sorts chromosome
value = value+1;phen=phen(ind,:);
%_____________________________________________
%iv. selection
prob=flipud([1:keep]'/sum([1:keep])); % weights of chromosomes
F=[0 cumsum(prob(1:keep))']; % probability distribution function
pick1=rand(1,M); % mate #1;pick2=rand(1,M); % mate #2
% mam and dad contain the indicies of the chromosomes that will mate
ic=1;
while ic<=M
for id=2:keep+1
if pick1(ic)<=F(id) & pick1(ic)>F(id-1);end
mam(ic)=id-1;
if pick2(ic)<=F(id) & pick2(ic)>F(id-1)
dad(ic)=id-1;end;end
```

```
ic=ic+1;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);
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
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);
%andphen(mutindx)= max([ub-
lb])*rand(1,Nmut)+min([lb]);%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('parto',phen(rowmut,:));
fout(keep+ii:Npop,:)=feval('parto',phen(keep+ii:Npop,:));
fout(keep+ii+1:Npop,:)=feval('parto',phen(keep+ii+1:Npop,:)
);
%_____________________________________________
%viii. Stopping criteria
if t>Generations
break ;end %if
%_____________________________________________
%ix. Displays the output
phen(1:10,:);[phen(elt,:) fout]
figure(1);plot(fout(:,1), fout(:,2),
'k.',f11,f21,'r*','LineWidth',3);xlabel('f1');ylabel('f2');grid on;%
plot pareto fronterif index of objective function is two
title('pareto fronter in objective space ');
figure(2);plot(phen(ind,1),phen(ind,2),'*');xlabel('x1');ylabel('
x2');grid on;
title('decision space'),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
```

6. Acknowledgment

First of all, I would like to express my deepest and special thanks to department of Mathematics, ASTU, and all members of the department are gratefully acknowledged for their encouragement and support. I would like to extend my thanks to Adama Science and Technology University for sponsorship and financial support. Lastly, I would like to forward my grateful thanks to my family for their incredible support and encouragement.

7. References

[1]. David A. Coley, An Introduction to Genetic Algorithms for Scientists and Engineers, 1999 by World Scientific Publishing Co. Pte. Ltd.

[2]. R. I. Bot, S-M. Grad, G. Wanka, Duality in Vector Optimization, Springer-Verlag Berlin Heidelberg (2009)

[3]. Deb, K., Multi-objective optimization using evolutionary algorithms, (2001), Wiley.

[4]. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. A Fast Elitist Non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, (2000).

[5]. M. Ehrgott: Multicriteria Optimization, Springer, 2005, New York

[6]. G. Eichfelder, Vector Optimization, Springer, 2008, Berlin Heidelberg

[7]. Fonseca, C. M. and Fleming, P. J,An overview of evolutionary algorithms in multiobjective optimization (1995).

[8]. C. J. Goh and X. Q. Yang, Duality in Optimization and Variational Inequalities, Taylor and Francis (2002), New York

[9]. Knowles, J. D. and Corne, D. W., The Pareto archived evolution strategy, A new baseline algorithm for multiobjective optimization, Proceedings of the 1999 Congress on Evolutionary Computation.

[10]. Kokolo, I., Kita, H., and Kobayashi, S, Failure of Pareto-based MOEAs: Does nondominated really mean near to optimal, Proceedings of the Congress on Evolutionary Computation 2001.

[11]. John R. Koza, Genetic Programming, 1992 Massachusetts Institute of Technology

[12]. Randy L. Haupt, Sue Ellen Haupt: practical genetic algorithms second edition, 2004 by John Wiley & Sons, Inc.

[13]. Y. Sawaragi, H. Nakayama and T. Tanino, Theory of Multiobjective optimization, academic press, 1985.

[14]. M. Semu (2003), On Cone d.c optimization and Conjugate Duality, Chin. Ann. Math, 24B:521-528.

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

[16]. S.N. Sivanandam, S.N. Deepa, Introduction to Genetic Algorithms, Springer-Verlag Berlin Heidelberg 2008

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