Non-dominated Rank based Sorting Genetic Algorithms

Ashish Ghosh
Machine Intelligence Unit and Center for Soft Computing Research
Indian Statistical Institute
203 B. T. Road, Kolkata 700108, India
ash@isical.ac.in

Mrinal Kanti Das
Indian Institute of Science
Bangalore, 560 012, India
nmrinl@csa.iisc.ernet.in

Abstract. In this paper a new concept of ranking among the solutions of the same front, along with elite preservation mechanism and ensuring diversity through the nearest neighbor method is proposed for multi-objective genetic algorithms. This algorithm is applied on a set of benchmark multi-objective test problems and the results are compared with that of NSGA-II (a similar algorithm). The proposed algorithm is seen to over perform the existing algorithm. More specifically, the new approach has been used to solve the deceptive multi-objective optimization problems in a better way.

Keywords: Multi-objective optimization, evolutionary computing, genetic algorithms, Pareto optimality

1. Introduction

Genetic Algorithm (GA) developed by *Holland* (1960) is a model of machine learning, that derives its behavior from a metaphor of the processes of *natural evolution* and *natural genetics*. Evolution, in essence, is a two-step process of random variation and selection. It can be modeled mathematically as:

$$X[t+1] = s(v(X[t]))$$

Address for correspondence: Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata 700 108, India, ash@isical.ac.in where the population at time t, denoted as $X[t]$, is operated on by random variation v, and selection s to give rise to a new population $X[t+1]$.

A GA uses variation and selection and the process goes on repeatedly on a population of candidate solutions/individuals. Individuals are encoded as strings called *chromosomes*, like nature holds all the basic information about an individual in the chromosomes. A GA is executed iteratively on this set of coded chromosomes, called a *population*, with the basic genetic operators: *selection* and *variation*. Variation, again, is a combination of *crossover* and *mutation* - two probabilistic operators. A GA uses only the objective function information (for selection) and probabilistic transition rule (for crossover and mutation). A GA works with the following basic genetic operators.

1.1. Basic genetic operators

1.1.1. Selection operator

The primary objective of the selection operator is to keep good solutions and eliminate bad solutions from a *population* keeping the population size (number of individuals in the population) constant, and progress towards a better collection of individuals from one generation to the next generation. This is achieved by performing the following tasks:

- Identify and select good (usually above average) solutions in a population.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

Popular ways to achieve the above tasks are **tournament selection**, **proportionate selection** and **ranking selection**.

In **tournament selection**, tournaments are played between two random solutions and the better solution is chosen and placed in the *mating pool* (mating pool is the collection of solutions for undergoing the operation of variation i.e. *crossover* and *mutation*). Two other solutions are picked up again and another slot in the mating pool is filled up with the better solution.

In the **proportionate selection** method, solutions are assigned copies the numbers of which are proportional to their *fitness values* (a numeric figure which signifies how fit a solution is in the competition for selection). If the average fitness value of population members is f_{avg}, a solution with a fitness value f_i gets an expected f_i/f_{avg} number of copies.

In **stochastic universal sampling (SUS)** only one random number r is chosen for the whole selection process. Since N different solutions have to be chosen, a set of N equal spaced numbers are created:

$$R = \{r, r + 1/N, r + 2/N, \ldots, r + (N-1)/N\} \; mod \; l,$$

where l is the total fitness value of all the solutions. Number of copies of solution i having fitness f_i is equal to the number of such numbers generated by the above technique in the range $[\ \sum_{j=1}^{i-1} f_j \quad \sum_{j=1}^{i} f_j\ ]$.

1.1.2. Crossover operator

In a natural process *crossover* occurs between pairs of chromosomes by exchanging parts of them. In a GA two strings are picked up randomly from the mating pool and some portions of them are exchanged to create two new strings. If a crossover probability of p_c is used, then $100 \times p_c\%$ strings are used in the crossover and $100 \times (1 - p_c)\%$ of the population is copied for possible mutation. Mutation operator follows crossover operator.

1.1.3. Mutation operator

In the natural process information about the creatures are stored in chromosomes in the form of genes. Each gene constitutes a behavioral aspect of that creature; mutation of a gene alters that particular aspect of the creature. In the same way each bit in the encoded chromosome of an individual solution in a particular GA process stores the information of a particular aspect of that solution. Changing that bit either from 0 to 1 or from 1 to 0 mutates it and eventually mutates the information encoded. Mutation is probabilistic in nature and helps to keep diversity in the population.

A GA with its basic genetic operators can be used to optimize a problem with one objective or multiple objectives. The process begins with a randomly selected initial population and progresses gradually towards the optimum solution. Since in this article, we will mainly be dealing with GA based multi-objective optimization, let us brief the concept of multi-objective optimization first.

2. Multi-objective optimization

A multi-objective optimization problem, as its name suggests, has a number of objectives that need to be optimized. One of the striking differences between single-objective and multi-objective optimization is that in multi-objective optimization one needs to take care of two spaces.

- Decision variable (search) space.
- Objective space.

Decision variable space is the domain of the candidate solutions. Its axes are the attributes/parameters of the candidate solutions. Objective space is the space where candidate solutions are projected through the objective functions and its axes are the objective functions of the problem. The objective functions map the candidate solutions from the decision variable space to objective space. Position of a candidate solution in the objective space determines how fit it is in the competition with other fellow candidate solutions for selection.

2.1. Criteria of multi-objective optimization

Real life problems consist of multiple objectives. In earlier attempts of finding solutions to these problems simpler single objective algorithms were used and a weighted average was considered. With this approach proper justification to all the objectives was not possible and hence the outcome in most of the cases remained far away from the actual solutions. To give importance to all the objectives of a problem, multi-objective approach is adopted. In this approach, obtaining a single perfect solution that simultaneously satisfy all the objective functions is impossible, generally a set of solutions known as Pareto-optimal solutions are obtained; and thus to have a proper trade-off among the solutions ensuring diversity became an important issue. So the basic objective is to provide proper justification to all the objectives in a real life problem and to have a good trade-off among the solutions obtained.

2.2. Pareto-optimal solutions

In multi-objective optimizing problems where the objective functions are conflicting in nature, each objective function may have a different individual optimal solution. So, to satisfy all objective functions, a set of optimal solutions is required instead of one optimal solution. The reason for the optimality of many solutions is that no one objective function can be considered to be better than any other. These solutions are non-dominated solutions. Let P be a set of non-dominated solutions. Then,

- Any two solutions of P must be non-dominated with respect to each other.
- Any solution not belonging to P is dominated by at least one member of P.

Actually, a solution x_1 is said to dominate another solution x_2 if both the following conditions are true:

- The solution x_1 is not worse than x_2 in all objectives, or $f_j(x_1)$ is better than $f_j(x_2)$ for all $j \in \{1, 2, \ldots, M\}$; where f_j is the j^{th} objective function and M is the number of objectives.
- The solution x_1 is strictly better than x_2 in at least one objective, or $f_j(x_1)$ is better than $f_j(x_2)$ for at least one $j \in \{1, 2, \ldots, M\}$.

Let us illustrate the Pareto optimality with "time & space complexity" of an algorithm shown in Figure 1. In this problem we have to minimize both time and space complexity. In multi-objective

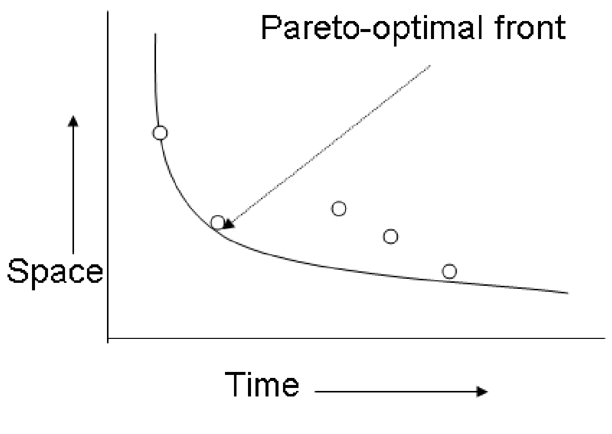

Figure 1. Pareto optimal solutions

optimization we have two goals:

- Progressing towards the Pareto-optimal front.
- Maintaining a diverse set of solutions in the non-dominated front.

Pareto-optimal front of a multi-objective optimization problem is the ideal solution set of that problem. In a GA we start with a random collection of candidate solutions, most likely they are not the best possible solutions and with the help of genetic operators we progress iteratively towards the best possible solution set after a number of generations. While progressing towards the Pareto-optimal set, it is also required to maintain a diverse set of solutions to provide good trade-off alternatives to the end user. There are various methods (briefed in Section 2.3) to ensure diversity among the obtained set of solutions.

2.3. Methods of ensuring diversity

2.3.1. Diversity through mutation

Mutation operator is often used as a diversity-preserving operator in GAs.

2.3.2. Pre-selection

Cavicchio (1970) was the first to introduce an explicit mechanism of maintaining diversity in a GA. Replacing an individual with a like individual is the main concept in a pre-selection operator. When an offspring is created from two parent solutions, the offspring is compared with the two parents. If the offspring has better fitness than the worst parent, it replaces that parent.

2.3.3. Crowding model

DeJong (1975) in his doctoral dissertation used a crowding model to introduce diversity among solutions in a GA population. In a crowding model, crowding of solutions anywhere in the search space is discouraged, thereby providing the diversity needed to maintain multiple optimal solutions. In this model only a proportion G (called *generation gap*) of the population is permitted to reproduce in each generation. Furthermore when an offspring is to be introduced in the overlapping population, CF (called the *crowding factor*) number of solutions from the population is chosen at random. The offspring is compared with these chosen solutions and the solution, which is most similar to that offspring, is replaced.

2.3.4. Sharing function model

Goldberg and Richardson (1987) suggested another concept, where instead of replacing a solution by a similar solution, the focus is more on degrading the fitness of similar solutions [11].

$$Sh(d_{ij}) = \begin{cases} 1 - (d_{ij}/\sigma_{share})^{\alpha} & \text{if } d_{ij} > \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

where d_{ij} is the distance between solution i and solution j, σ_{share} is the sharing parameter, α indicates the degree of sharing, $Sh(d_{ij})$ is the sharing value between solutions i and j.

Then, $Nc_i(= \sum_{j=1}^{N} Sh(d_{ij}))$ is the niching value of solution i. The modified fitness value $f_i' = f_i/Nc_i$.

2.3.5. Hyper-cube model

In this model each objective is divided into 2^d equal divisions, where d is a user-defined depth parameter. In this way, the entire search space is divided into $(2^d)^M$ unique, equal-sized M-dimensional hyper-cubes, where M is the number of objectives. The archived solutions are placed in these hyper-cubes according to their locations in the objective space. If the offspring resides in a less crowded hyper-cube than the parent, the offspring becomes the parent of the next generation. Otherwise the parent solution continues to be the parent of the next generation [15].

3. Algorithms available for multi-objective optimization

Many researchers working on multi-objective optimization problems have suggested various algorithms using genetic algorithms. These algorithms are more or less similar in utilizing the basic genetic operators like selection, crossover, mutation, elite preservation operator (wherever applicable); they actually differ in the way the fitness is assigned to individuals. Various algorithms are available [5, 7, 13, 14, 15, 16, 18, 19, 22, 23, 24, 26, 28, 30, 31] to solve multi-objective optimization problems. Here is a brief discussion on some of the standard multi-objective genetic algorithms.

3.1. Non-elitist models

3.1.1. Vector evaluated genetic algorithm (VEGA) [26]

The population is divided (at every generation) into a number of equal sub-populations randomly. All the solutions in a sub-population are assigned a fitness value based on one objective function, and each of the objective functions is used to evaluate members in the population, i.e. each sub-population is evaluated on one objective function. Fitness proportionate selection is done in each subpopulation, and the selected elements of all the sub-populations are aggregated to make the next mating pool. The above procedure is repeated until convergence.

3.1.2. Weight-based genetic algorithm (WBGA) [13]

Each objective function f_i is multiplied by weight w_i. The weighted objective functional values are then added together to calculate the fitness of a solution. Random weighted GA (RWGA) [22] is similar to WBGA, except that a random normalized weight vector $w^{(i)} = (w_1, w_2, , w_M)^T$ is assigned to the solution i. The fitness value of the solution is calculated as the weighted sum of the objectives with $\sum_{j=1}^{M} w_j = 1$, where M is the number of objectives. $f(x^{(i)}) = \sum_j w_j^{(i)} f_j(x^{(i)})$. Once the fitness values are determined, we continue with selection and reproduction operations of standard genetic algorithms.

3.1.3. Multiple objective genetic algorithm (MOGA) [7]

It is the first algorithm, which uses the non-dominated classification of a population. Each solution is checked for its domination in the population and a rank i, equal to one plus the number of solutions n_i that dominates solution i, is assigned. In order to ensure diversity this algorithm uses *sharing model*.

3.1.4. Non-dominated sorting genetic algorithm (NSGA) [10, 28]

In NSGA the population is sorted according to non-domination and classifies the population into a number of mutually exclusive equivalent classes P_j. If P is the whole population, then $P = \bigcup_{j=1}^{p} P_j$.

The *sharing function model* is used to ensure diversity. It is used front-wise and ensures that the fitness values of all the solutions of front f are less than that of any solution of front $f - 1$.

3.1.5. Niched-Pareto genetic algorithm (NPGA) [14]

This method differs from the previous methods in selection operator. It uses the tournament selection, unlike the proportionate selection used in VEGA, NSGA and MOGA. One attractive aspect of NPGA is that there is no need for specifying any particular fitness value to each solution. The tournament selection prefers non-dominated solutions in a stochastic manner.

3.1.6. Predator-prey evolution strategy [18]

This algorithm does not use a domination check to assign fitness to a solution. Instead the concept of a predator-prey model is used. Preys represent a set of solutions $(x^{(i)}, i = 1, 2, , N)$, that are placed on the vertices of an undirected connected graph. First, each predator is randomly placed on any vertex of the graph. Each predator is associated to a particular objective function. Secondly, staying on a vertex, a predator looks around for preys in its neighboring vertices. The predator catches a prey having the worst value of its associated objective function. When a prey x^i is caught, it is erased from the vertex and a new solution is obtained by mutating (and recombining) a random prey in the neighborhood of x^i, the new solution is then kept on the vertex. After this event is over, the predator takes a random walk to any of its neighboring vertices. The simultaneous presence of predators favoring each objective allows trade-off solutions to co-exist in the graph.

3.2. Elitist algorithms

3.2.1. Elitism

Elitism is a useful concept to accelerate the process of obtaining the final optimal set of solutions by preserving the good solutions already found. In a simple implementation, the best $\epsilon\%$ of the population from the current population is directly copied to the next generation. The rest $(100-\epsilon)\%$ of the new population is created by the usual genetic operations applied on the entire population. In another implementation, two offspring are compared with two parent solutions and two better solutions are preserved.

3.2.2. Rudolph's elitist multi-objective evolutionary algorithm [24]

In its general format, m parents are used to create l offspring using genetic operators. Now there are two populations: the parent population P_t and the offspring population Q_t. The algorithm works in three phases. In the first phase, the best solutions in Q_t (i.e. non-dominated solutions) are identified and moved from Q_t to Q^*. In the second phase each solution q of Q^* is compared with each solution of the parent population P_t. If q dominates any solution of P_t, that solution cannot be an elite solution in accordance with Q^* and is thus deleted from P_t. Since this offspring is special (at least it has dominated one parent solution), it is taken out of Q^* and put in a set P. On the other hand, if q does not dominate any solution of P_t and any solution in P_t do not dominate q, then q belongs to the same non-dominated set as all solutions of P_t. Such a solution q is also special to a lower degree than the elements of P in the sense that at least q does not get dominated by any parent solution of P_t, it is taken out of Q^* and put into another set Q. In the third phase, all the above sets are arranged in a special order of preference to fill up the next generation. N solutions from these sets are taken out in the order P_t, P, Q, Q^*, Q_t, where N is the population size.

3.2.3. Distance-based Pareto genetic algorithm [23]

This algorithm emphasizes the progress towards the Pareto-optimal front and the diversity along the obtained front by using one fitness measure. The first population P_0 of size N is created at random and all the solutions are assigned random fitness values and are automatically added to the elite set E_0. Thereafter, each solution is assigned a fitness based on its average distance from the elite set. The archive of the elite solutions is updated with the non-dominated set of solutions taking into consideration the offspring population and the already existing elite solutions.

3.2.4. Strength Pareto evolutionary algorithm (SPEA) [31]

The algorithm begins with a randomly created population P_0 of size N and an empty external population P_0 with a maximum capacity of N. SPEA assigns a fitness (called *strength*) S_i to each member i of the external population first. The strength S_i is $n_i/(N+1)$, where n_i is the number of solutions of the current population which are dominated by external solution *i*. Then the fitness F_j of an individual j is assigned as

$$F_j = 1 + \sum_{i \in P_t' \ AND \ i \leq j} S_i.$$

It is worth noting that a solution with smaller fitness is better. Archive is maintained to preserve elite solutions. Clustering is used to maintain the size of the archive of the elite solutions and to ensure diversity among the elite solutions.

3.2.5. Elitist non-dominated sorting genetic algorithm (NSGA-II) [5]

In this algorithm, parent population P_t of size N and offspring population Q_t of size N are combined to form R_t of size $2N$. Then a non-dominated sorting is used to classify the entire population R_t. The new population is filled by solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front and continues with solutions of the second non-dominated front, followed by the third non-dominated front, and so on. Since the overall size of R_t is $2N$, all the fronts may not be accommodated in N slots available in the new population. All the fronts, which could not be accommodated, are deleted. When the last allowed front is being considered, there may exist more number of solutions in that front than that required to fill up the remaining slots of the new population. So, some solutions would be discarded from that front and other solutions would be selected. Instead of arbitrarily discarding solutions, a niching strategy is used to choose the solutions, which reside in the least crowded region in that front, to be selected in the new population. As a niching strategy, crowded tournament selection operator [5] is used.

The main distinguishing advantage of this algorithm is that while ensuring diversity no extra niching parameter is required. The elitist mechanism does not allow an already found Pareto-optimal solution to be deleted. But when the crowded comparison is used to maintain the population size, convergence to the Pareto-optimal front is hindered. Another flaw is, if the number of solutions in the first front does not exceed the population size, diversity among the finally obtained set may not be properly ensured. So, if in any problem it fails to converge to actual optimal set, it will also fail to provide good trade-off among the obtained set of solutions.

In the following section we propose a multi-objective GA to overcome these limitations.

4. Non-dominated rank based sorting genetic algorithm (NRSGA)

The basic concept of this algorithm is to classify the population into fronts first and then assigning ranks to the solutions in a front. Ranking is assigned with respect to all the solutions of the population and each front is reclassified with this ranking. Diversity is ensured using a simple and direct nearest neighbor distance approach. To ensure elitism the parent population and the offspring populations together participate in the competition and the best set of solutions from them are selected for next generation. The distinguishing features of this algorithm are:

- Reclassifying the solutions of the same front on the basis of ranks.
- Successfully avoiding sharing parameters though ensuring diversity among trade off solutions using the nearest neighbor method.

4.1. Procedure

In the following text we will describe the proposed algorithm using examples of a number of minimization problems. For solving maximization problems, they need to be converted into minimization problems.

First, classify the whole population into fronts on the basis of non-dominance. Put the non-dominated solutions in the first front. To determine the elements of the next front, discard the elements of the first front from consideration and find out the non-dominated solutions from the rest of the populations and put them in the second front. In the same way neglecting the solutions of the first and the second front, the non-dominated solutions of the rest of the population are put in the third front. Repeat this process until each solution belongs to a front.

Next, assign ranks to the solutions. The rank of solution i is $r_i = n_i + 1$, where n_i is the number of solutions those dominate solution i.

Then, adjust the fitness values of the solutions. Assign fitness values to all the solutions of the population depending on their position in the fronts. The average fitness value assigned to the solutions of the first front is the number of solutions in the population. Thus $F_{avg} = N$ for solutions of the first front.

Adjust the average fitness values on the basis of their ranking. If F_{avg} be the average fitness value of solution i, and F_{adj} be the adjusted fitness value of it, then $F_{adj} = F_{avg} - g$; where g is the number of solutions (in the same front of solution i) having rank less than or equal to that of solution i. Then the density factor is considered, and the actual fitness value is defined as: $F = F_{adj} - 1/Min$; where *Min* is the distance between solution i and its nearest neighbor. Proper care (rescaling etc.) must be taken to keep the fitness value positive for all elements of all fronts.

If F_j be the minimum fitness of some solution j and $F_j < 0$, then add $(0 - F_j)$ to the fitness of all the solutions to make them positive.

Next, find out the minimum of the fitness values of the first front, and the average fitness values to the solutions of the second front are assigned as $F_{avg}(ii) = F_{avg}(i) - \epsilon$, where, ϵ is a small positive value.

Repeat the above two steps to assign fitness values to the solutions of rest of the fronts.

In this approach it is ensured that the fitness values of all the solutions of front f is less than that of any solution of front $f - 1$. Between the convergence property and diversity, the first one is given more priority.

Maintaining elitism: From the second generation elitism is considered. The solutions of the current population and previous population are merged to form a population of size $2N$. Then the fitness values are assigned to all the solutions according to the previously described rule. The combined population is sorted according to the fitness values and best N solutions are selected to advance to the next generation. So in this way the parent solutions and the offspring solutions are allowed to compete together and the best N numbers of solutions are picked up from them.

Let us describe the above process in the form of an algorithm. (Please note that the algorithm to formulate the whole process is quite large to accommodate in this place. Hence the main points have been presented.)

4.2. Algorithm

4.3. Advantages and disadvantages

The proposed algorithm ensures diversity with no extra niching parameter. Elitism does not allow an already found Pareto-optimal solution to be deleted. As ensuring diversity is parameterized implicitly in the fitness criteria, convergence property does not get affected while ensuring diversity; and to ensure diversity, distances between nearest neighbors are considered which is fast and simple.

The algorithm has an extra-added complexity for reclassifying the solutions of the same front based on ranks.

5. Implementation and results

The algorithm is implemented and compared with NSGA-II (most similar, among the existing algorithms, to the proposed one) using the following benchmark problems. Population size N was chosen as 100, for all problems 50 simulations were made, crossover probability p_c and mutation probability p_m were considered as 0.8 and 0.01 respectively. In the tables we put the average results of 50 runs. Pareto set cardinality was100, $\sigma_{share} = 0.158, and\epsilon = 0.5$.

Algorithm for classifying into fronts

```
For each solution $i$ in population $P$ /* initialization
Fronts$[i] = 0$
Front$= 0$
While( true)
Start:
Front$=$Front$+1$
For each solution $i$ in population $P$
If Fronts$[i] = 0$ then /* The solution is not assigned a front */
For each solution $l$ in population $P$
If Fronts$[l] = 0$ or Fronts$[l]$ =Front then
If for each objective function $j$
$OPop[l][j] >= OPop[i][j]$ /* solution $i$ dominates solution $l$ */
Fronts$[i]$ =Front /* solution $i$ is not dominated by any solution which is
not assigned a front or belongs to the same front*/
End if
End of loop $l$
End if
End of loop $I$
If any element remains to be checked go to Start
Else Break
End if
End while
```

Algorithm for ranking

```
For each solution $i$ in population $P$
Rank= 1 /* ranking starts with 1 (one) */
For each solution $l$ in population $P$
If $l \neq i$ then
Flag=false
For each objective function j
If $OPop[l][j] <= OPop[i][j]$ then /* solution $l$ is not dominated by solution $i$*/
Continue /* until it finds some solution which dominates solution $l$, continue*/
Else
Flag=true
Break
End if
End of loop $j$
If flag=false then
Rank=Rank+1 /* rank gets increased by one, the number of times it finds a
            solution which dominates solution $l$ */
End if
End of loop $l$
Ranks[$i$] =Rank
End of loop $I$
```

Algorithm for fitness adjustment

```
/* each solution is assigned a fitness value initially */
For each solution $i$ in population $P$
If Fronts$[i]$ =Front then /* considering solutions front by front */
Sum= $0$
For $j = 1$ to Ranks$[i]$ increment $1$
For each solution $k$ in population $P$
If Ranks$[k] = j$ AND Fronts$[k]$ =Front then
Sum=Sum$+1$ /* value of 'Sum' gets increased the number of solutions have rank less than
            that of solution $i$ */
End if
End of loop $k$
End of loop $j$
FitnessValue$[i]$ =FitnessValue$[i]-$Sum
Let $D =$ the distance between solution $i$ and its nearest neighbor in the Population $P$
/* $D$ is calculated by the method described in section 2.3.4 */
FitnessValue$[i]$ =FitnessValue$[i] - 1/D$.
```

Fitness assignment scheme

```
/* here goes the summary */
1. Classify the whole population $P$ into Fronts.
2. Classify the whole population $P$ into Ranks.
3. Assign the fitness value to each solution of a front considering front, rank and
the nearest neighbor distance parameters.
4. Adjust all the solutions if $min$ is less than zero, by adding a value equal to $zero - min$,
where $min$ is the minimum fitness value of all solutions in the previous front.
5. Assign average fitness value to all the solution of next front Favg=min.
6. Repeat steps 3 and 4 until all the solutions in population $P$ are checked.
```

5.1. The benchmark test problems

$$SCH1: \begin{cases} Minimize & f_1(x) = x^2 \\ Minimize & f_2(x) = (x-2)^2 \\ & -A \leq x \leq A. \end{cases}$$

It is a single variable test problem with a convex Pareto optimal set [25].

$$SCH2: \begin{cases} Minimize & f_1(x) = \begin{cases} -x & \text{if } x \leq 1 \\ x-2 & \text{if } 1 < x \leq 3 \\ 4-x & \text{if } 3 < x \leq 4 \\ x-4 & \text{if } x > 4 \end{cases} \\ Minimize & f_2(x) = (x-5)^2 \\ & -5 \leq x \leq 10. \end{cases}$$

The Pareto optimal set consists of two discontinuous regions. The main difficulty that an algorithm may face in solving this problem is that a stable subpopulation on each of the two disconnected Pareto optimal regions may be difficult to maintain [25].

$$FON: \begin{cases} Minimize & f_1(x) = 1 - \exp(-\sum_{i=2}^{n}(x_i - 1/\sqrt{n})^2) \\ Minimize & f_1(x) = 1 - \exp(-\sum_{i=2}^{n}(x_i + 1/\sqrt{n})^2) \\ & -4 \leq x_i \leq 4, \quad i \in \{1, 2, \ldots, n\}. \end{cases}$$

The Pareto optimal set is non convex [8].

$$ZDT1: \begin{cases} f_1(x) = x_1 \\ g(x) = 1 + 9/(n-1)\sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - \sqrt{(}f_1/g). \end{cases}$$

It has a continuous Pareto optimal front and a uniform distribution of solutions across the front. The only difficulty with this problem is in tackling a large number of decision variables [33].

$$ZDT2: \begin{cases} f_1(x) = x_1 \\ g(x) = 1 + 9/(n-1)\sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - (f_1/g)^2. \end{cases}$$

In the Pareto optimal region it has a uniform density of solutions. The difficulty with this problem is that the Pareto optimal region is non convex [33].

$$ZDT3: \begin{cases} f_1(x) = x_1 \\ g(x) = 1 + 9/(n-1)\sum_{i=2}^{n} x_i \\ h(f_1, g) = 1 - \sqrt{(}f_1/g) - (f_1/g)\sin(10\pi f_1). \end{cases}$$

Difficulty with this problem is that the Pareto optimal region is discontinuous. The real challenge for an algorithm would be to find all discontinuous regions with a uniform spread of non-dominated solutions [33].

$$ZDT4: \begin{cases} f_1(x) = x_1 \\ g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)) \\ h(f_1, g) = 1 - \sqrt{(f_1/g)}. \end{cases}$$

The sheer number of multiple local Pareto optimal fronts produces a large number of hurdles for an algorithm to converge to the global Pareto optimal front [33] for the problem.

Some standard metrics, described below, are used to evaluate the algorithms on these benchmark test problems.

5.2. Metric for comparison

5.2.1. To measure convergence property

General distance: It is a measure of general distance between the obtained front and the ideal Pareto optimal front. Instead of finding whether a solution Q belongs to the set P^* (Pareto set) or not, this metric finds an average distance of this solution from P^*. For $p = 2$, the parameter d_i is the Euclidean distance (in the objective space) between a solution $i \in Q$ and the nearest member of P^*. f_m^k is the m^{th} objective function value of the k^{th} member of P^*. Lesser the value, better is the performance [30].

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|} \text{ with}$$

$$d_i = \min_{k=1}^{|P^*|} \sqrt{(}\sum_{m=1}^{M} (f_m^i - f_m^{*k})^2.$$

5.2.2. To measure diversity among the solutions in the obtained front

Spacing: It is a measure of average spacing among the solutions in the obtained front. This is calculated with a relative distance measure between consecutive solutions in the obtained non-dominated set as follows:

$$S = \sqrt{(\frac{1}{|Q|} \sum_{i=1}^{|Q|} (d_i - \bar{d})^2)}$$

where

$$d_i = \min_{K \epsilon Q \bigwedge K = i} \sum_{m=1}^{M} |f_m^i - f_m^k|$$

$$\bar{d} = \sum_{i=1}^{|Q|} \frac{d_i}{|Q|}$$

and $\bar{d}$ is the mean value of these distances.

This distance measure is the minimum value of the sum of the absolute differences in the objective functional values between the i^{th} solution and any other solution in the obtained non-dominated set. The above metric measures the standard deviations of different d_i values. When the solutions are mostly uniformly spaced the corresponding distance measure will be small. The lesser the value, the better is the result [27].

Spread: It signifies how much the solutions are spread in the obtained front in the search space, lesser the value, the better is the result [5].

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} (d_i - \bar{d})^2}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}}.$$

It is measured to determine the spread of solutions. Solutions with good spacing imply that they are almost uniformly spaced (among them) but may be clustered in a small place over the front. In that case spread will be poor.

5.3. Results

The results obtained in the form of standard metric when applied on the benchmark test problems are listed below for different problems.

Table 1. **SCH1**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	148462	6	0.00009999
NRSGA	69504	8492	0.00001211

Table 2. **SCH2**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	42.002164	0.0000766	0.00009969
NRSGA	0.003967	0.0000899	0.00008110

6. Discussion of results

In the tables we have seen a statistical report on the performance of the algorithms. Now let us discuss the similarities-dissimilarities and advantages-disadvantages of these two algorithms. Both of these

Table 3. **FON**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	0.0052	0.0000000	0.00009999
NRSGA	0.0022	0.0000964	0.00003621

Table 4. **ZDT1**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	0.001330	0.000037	0.00009528
NRSGA	0.001202	0.00001431	0.00005641

Table 5. **ZDT2**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	0.001665	0.0000188	0.00007837
NRSGA	0.001360	0.00001806	0.00004920

Table 6. **ZDT3**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	0.002053	0.000082	0.00009478
NRSGA	0.001875	0.0000555	0.00007121

Table 7. **ZDT4**

Algorithms	**General distance**	**Spacing**	**Spread**
NSGA-II	1.006070	0.00001886	0.00006182
NRSGA	0.009386	1.00004195	0.00003858

algorithms are multi-objective optimization algorithms incorporated with basic genetic operators like *selection*, *crossover*, *mutation* and *elitism*. Except all these basic genetic operators there lies an important point - fitness assignment to a solution i.e. to judge how fit a solution is in the search space than the other solutions in a problem definition and how quick they converge to the optimum solution set.

At first both of these algorithms classify the whole search space into collection of fronts on the basis of non-dominance. Totally non-dominated solutions lie in the first front, and in the second front the solutions are dominated only by the solutions of the first front. Likewise solutions of the third front are dominated only by the solutions of the first and the second fronts. At the next step NRSGA reclassify the solution space giving each solution n_i a rank $r_i = n_i + 1$, n_i is the number of solutions that dominate solution n_i. The main distinction between these two algorithms is in this *ranking* concept. NRSGA uses the concept of ranking among the solutions of the same front; NSGA-II does not do so. The main motivation behind the ranking concept is to reclassify the solutions of the fronts. By selecting the solutions of the first front we select the best possible solutions from the perspective of convergence. The solutions of the first front would have the same rank, but solutions of other fronts would have different ranks. Rank of a solution is less if it is dominated by less number of solutions of lower fronts. This can be described in another way. If the whole population is subdivided with a number of equidistant parallel lines slanting normally to the Pareto optimal front, then solutions with less rank in any front would fall in the part where comparatively less number of solutions is there altogether. In this way diversity among solutions also increases. The concepts of ranking and fitness reallocation using the nearest neighbor method reduce the probability of picking up solutions from the densely populated place.

This is to be mentioned here that the concept of ranking is not new. Bentley has used the concept of ranking earlier, but his approach is different from that of ours. Bentley used the method of ranking introducing the concept of *'importance'*, that is, rank was imposed on the solutions giving importance to particular objective. Pareto optimal solutions are chosen based on these ranks. But in our method we give importance to all the objectives equally. This method of ranking is not to choose a subset of Pareto optimal solutions which is more eligible than other solutions within the perspective of any particular important objective. Ranking is used in this method to reclassify the whole set of solutions based on dominance to accelerate the process to converge to Pareto optimal front, and also to avoid the solutions to be localized. This again effectively enhances the probability of preserving diverse solutions. So the problems with deceptive nature are solved in a better way with this reclassification concept in most of the cases. But, if the ranks of the solutions of the same front are the same in most of the cases, then the reclassification is not much useful. So for complex problems where search space is not smooth and multimodal, the objective functions are deceptive in nature, the proposed algorithm will be more useful.

In elite preserving operator both the algorithms follow the same policy. They merge parent and offspring populations and assign fitness values to all the solutions of the combined population and select the N best possible solutions from the $2N$ number of candidate solutions.

In case of NSGA-II, the population is classified into groups of fronts. The solutions of the first front are selected if the number of solutions is less than N (population size), then the solutions of the second front is selected (if the number of solutions selected does not exceed N). This goes on until for front F_k, the number of solutions in F_k is more than the number of solutions required to fill the N slots of the selected candidate solutions. Then Crowded Tournament Selection Operator (CTSO - discussed earlier) [5] is used to select exactly the number of solutions that are required to fill the remaining slots. In case of NRSGA, each solution, in addition to the front has two more parameters (information about their dominance and position in the search space) - the first one is the rank value (how many solutions dominate it) and the second is the distance value (distance between it and its nearest neighbor). All these information collectively evaluate the fitness value of a solution; but it is ensured that the dominance information (the parameter of convergence to the Pareto-optimal front) is given higher priority than the distance information (which is the parameter of diversity maintenance among the finally obtained solution set). It is also ensured that a solution of front f has better fitness value than a solution of front $f + 1$. Then with the fitness values available for all the solutions, Stochastic Universal Sampling (SUS discussed earlier) [4] is used to select the required number of solutions.

So in the *selection* procedure, NRSGA uses SUS and NSGA-II uses CTSO. As CTSO is a binary tournament selection operator it operates in $O(N \log N)$ (N: population size); whereas SUS operates in $O(N)$ time complexity. Moreover in NRSGA in the fitness value both the fitness (i) fitness based on its non-dominance in the search space, (ii) fitness based on diversity measures, are combined in a single numerical value.

Another striking distinction between these two algorithms is in ensuring diversity among the solutions of Pareto-optimal front. In NSGA-II no exclusive method is adopted to select the lonely solutions (i.e. solutions in comparatively less dense zone). The lonely solutions are given priority in the situation of tie breaking i.e. when the last front to be included in the final population and the size of the front is larger than the number of solutions required then the required no of solutions are selected on the basis of their loneliness. But in NRSGA the two principal objectives of multi-objective optimization problems, diversity and convergence are taken into consideration simultaneously with convergence given higher priority. Both the parameters are encoded in the fitness value, thus a solution i has better fitness value than solution j means solution i is better in both the aspects.

This is also seen from the tables that NRSGA has come out with better results in diversity factor in most of the problems and what is more important is that diversity is ensured without degrading the convergence factor.

Both NRSGA and NSGA-II uses crossover and mutation in the same way; single point crossover with probability 0.8 and mutation with probability 0.04 were used for the present experiments.

ZDT4 causes problem for many standard algorithms in finding an optimal front close to Pareto optimal front due to its deceptive nature with multiple local Pareto-optimal fronts. The reclassification property of NRSGA helps it to achieve convergence; and its results are better in comparison with NSGA-II in ZDT4 also.

The Pareto-optimal front in SCH2 consists of two discontinuous regions; in ZDT3 the Pareto-optimal front is discontinuous and in FON and ZDT2 the Pareto-optimal front is non-convex; the reclassification of the fronts and usage of extra information of the solutions regarding their rank in the population helps to come out with better results.

Many standard algorithms faced problem while solving ZDT3 to have uniformly spread solutions in Pareto-optimal front, but NRSGA has overcome this difficulty in a better way. In comparison with NSGA-II, results are comparable but NRSGA provides better results in most of the cases in both the aspects (i.e. convergence and diversity).

In some problems both the algorithms give good and comparative results. As in SCH1 NRSGA has shown better performance to converge to ideal Pareto-optimal set; whereas NSGA-II is better in providing good diversity among the obtained set of solutions. For SCH2, NRSGA is better to converge to ideal Pareto-optimal set but in providing diversity, both the algorithms are comparable and good. For FON both the algorithms are almost similar in outcome and NRSGA has less *general distance* value. For ZDT2 both the algorithms give good results and the results are quite comparable in all the metrics.

7. Conclusion

In this paper a new concept, ranking among the solutions of the same front, has been proposed. It implements elitism and ensures diversity through the nearest neighbor method. This algorithm is applied on seven benchmark multi-objective test problems and the results are compared with that of NSGA-II. In most of the cases NRSGA has come out with better results in both the aspects of multi-objective optimization (i.e. convergence and diversity). It is more interesting to note that NRSGA is stochastically more reliable for deceptive problems. But the main disadvantage of NRSGA is its extra complexity due to reclassification. In future we plan to find out a better approach to reduce this complexity and we intend to apply NRSGA to solve other multi-objective problems.

References

[1] Bentley, P. J. and Wakefield, J. P. (1997). Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms.

[2] Cavicchio, D. J. (1970). Adaptive Search Using Simulated Evolution. Ph. D. Thesis, Ann Arbor, MI: University of Michigan.

[3] Coello, C. A. (2000). Handling preferences in evolutionary multiobjective optimization: a survey. In 2000 Congress on Evolutionary Computation, Vol. 1, pp. 30-37.

[4] Deb, K. Multi-Objective Optimization using Evolutionary Algorithms. Chichester, UK: Wiley, 2001

[5] Deb, K., Agarwal, S., Pratap, A. and Meyarivan, T. (2000b). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Proceedings of Parallel Problem Solving from Nature VI (PPSN-VI), pp. 849-858.

[6] DeJong, K. A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph. D. Thesis, Ann Arbor, MI: University of Michigan.

[7] Fonesca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416-423.

[8] Fonesca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multi-objective optimization. Evolutionary Computation Journal 3(1), pp. 1-16.

[9] Fonesca, C. M. and Fleming, P. J. (1996). On the performance assessment and comparison of stochastic multi-objective optimizers. In Proceedings of Parallel Problem Solving from Nature IV (PPSN-IV), pp. 584-593.

[10] Goldberg, D. E. (1989). Genetic Algorithms for Search, Optimization and Machine Learning. MA: Addison-Wesley.

[11] Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 41-49.

[12] Hajela, P. and Lin, C. Y. (1992). Genetic search strategies in multi-criterion optimal design. Structural Optimization 4(2), pp. 99-107.

[13] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: MIT Press.

[14] Horn, J., Nafploitis, N. and Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimization. In Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 82-87.

[15] Knowles, J. D. and Corne, D. W. (2000). Approximating the non-dominated front using the Pareto archived evolution strategy. Evolutionary Computation Journal 8(2) , pp. 149-172.

[16] Krishnakumar, K. (1998). Micro-genetic algorithms for stationery and non-stationery function optimization. In SPIE Proceedings: Intelligent Control and Adaptive Systems, pp. 289-296.

[17] Kursawe, F. (1990). A variant of evolutionary strategies for vector optimization. In Parallel Problem Solving from Nature I (PPSN-I), pp. 193-197.

[18] Laumanns, M., Rudolph, G. and Schwefel, H. P. (1998). A spatial predator-prey approach to multi-objective optimization: a preliminary study. In Proceedings of the Parallel Problem Solving from Nature V (PPSN-V), pp. 241-249.

[19] Leung, K. S., Zhu, Z. Y., Xu, Z. B. and Leung, Y. (1998). Multiobjective optimization using non-dominated sorting in annealing genetic algorithms. Department of Geography and Centre for Environmental Studies, Chinese University of Hong Kong, Hong Kong.

[20] Lirsawe, F. (1991). A variant of evolution strategies for vector optimization. In Parallel Problem Solving from Nature I (PPSN I), volume 496 of Lecture Notes in Computer Science, pp. 193-197.

[21] Murata, T. and Ishibuchi, H. (1995). MOGA: multi-objective genetic algorithms. In Proceedings of the Second IEEE International Conference on Evolutionary Computation, pp. 289-294.

[22] Neef, M., Thierens, D. and Arciszewski, H. (1999). A case study of a multiobjective recombinative genetic algorithm with coevolutionary sharing. In Proceedings of the Congress on Evolutionary Computation (CEC - 1999), pp. 796-803.

[23] Osyczka, A. and Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Structural Optimization 10(2), pp. 94-99.

[24] Rudolph, G. (2001). Evolutionary search under partially ordered fitness sets. In Proceedings of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ISI 2001), pp. 818-822.

[25] Schaffer, J. D. (1984). Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms. Ph. D. Thesis, Nashville, TN: Vanderbit University.

[26] Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In proceedings of the first International Conference on Genetic Algorithms, pp. 93-100.

[27] Schott, J. R. (1995). Fault Tolerant Design Using Single and Multi-Criteria Genetic Algorithms. Master's Thesis, Boston, MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

[28] Srinivas, N. and Deb, K. (1994). Multi-objective function optimization using non-dominated sorting genetic algorithms. Evolutionary Computation Journal 2(3), pp. 221-248.

[29] Test problems for multiobjective optimizers (Eckart Zitzler's page). http://www.tik.ee.ethz.ch/ zitzler/testdata.html.

[30] Veldhuizen, D. V. (1999). Multiobjective Evolutionary Algorithms: Classifiation, Analyses, and New Innovations. Ph. D. Thesis, Dayton, OH: Air Force Institute of Technology. Technical Report No. AFIT/DS/ENG/99-01.

[31] Zitzler, E. and Thiele, L. (1998a). An evolutionary algorithm for multiobjective optimization: The Strength Pareto approach. Technical Report 43, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology.

[32] Zitzler, E. (1999). Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph. D. Thesis, Zurich, Switzerland: Swiss Federal Institute of Technology.

[33] Zitzler, E., Deb, K. and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computational Journal 8(2), pp. 125-148.