Evolutionary Algorithms for Multi-Criterion Optimization: A Survey

Ashish Ghosh
Machine Intelligence Unit, Indian Statistical Institute, Kolkata-108, India.

Satchidananda Dehuri
Department of Computer Science & Engineering,
Silicon Institute of Technology, BBSR-24, India.

Abstract: *In this paper, we review some of the most popular evolutionary algorithms and a systematic comparison among them. Then we show its importance in single and multi-objective optimization problems. Thereafter focuses some of the multi-objective evolutionary algorithms, which are currently being used by many researchers, and merits & demerits of multi-objective evolutionary algorithms (MOEAs). Finally, the future trends in this area and some possible paths of further research are addressed.*

Keywords: *Evolutionary algorithms, multi-criterion optimization, Pareto-optimal solution, dominated and non-dominated solution.*

Received: *March 01, 2004* | **Revised:** *February 03, 2005* | **Accepted:** *March 10, 2005*

1. Introduction

Evolutionary techniques have been used for the purpose of single-objective optimization for more than three decades [1]. But gradually people discovered that many real world problems are naturally posed as multi-objective. Now a day's multi-objective optimization is no doubt a very popular topic for both researchers and engineers. But still there are many open questions in this area. In fact there is not even a universally accepted definition of "optimum" as in single objective optimization, which makes it difficult to even compare results of one method to another, because normally the decision about what the "best" answer is corresponds to the so called (human) decision-maker.

Since multi-criterion optimization requires simultaneous optimization of multiple often competing or conflicting criteria (of objectives), the solution to such problems is usually computed by combining them into a single criterion optimization problem. But the resulting solution to the single objective optimization problem is usually subjective to the parameter settings chosen by the user [2], [3]. Moreover, since usually a classical optimization method is used, only one solution (hopefully a Pareto-optimal solution) can be found in one simulation run. So, in-order to find multiple Pareto-optimal solutions, evolutionary algorithms are the best choice, because it deals with a population of solutions. It allows to finding an entire set of Pareto-optimal solutions in a single run of the algorithm. In addition to this, evolutionary algorithms are less susceptible to the shape or continuity of the Pareto front.

The rest of the paper is organized as follows. Section 2 gives a brief description of evolutionary algorithm. Section 3 addresses the key concept of multi-objective evolutionary algorithm. The different approaches of MOEA and their merits and demerits are the subject of section 4. A discussion of the future perspectives is given in section 5. Section 6 concludes the article. Some applications of multi-objective evolutionary algorithms are also indicated.

2. Evolutionary Algorithms

Evolution is in essence a two-step process of random variation and selection [4]. It can be modeled mathematically as $x[t+1] = s(v(x[t]))$, where the population at time t, denoted as $x[t]$, is operated on by random variation v, and selection to give rise to a new population $x[t+1]$. The process of evolution can be modeled algorithmically and simulated on a computer. The modeled algorithm is known as *evolutionary algorithms (EAs)*. *Evolutionary computation* is the field that studies the properties of these algorithms and similar procedures for simulating evolution on a computer. The subject is still relatively new and represents an effort to bring together researchers who have been working in these and other closely related fields.

In essence, the EAs are used to describe computer-based problem solving algorithms that use computational models of evolutionary processes as key elements in their design and implementation. A variety of evolutionary algorithms have been evolved during the last 30 years. The major ones are i) *Genetic algorithms*, ii) *Evolution strategies*, iii) *Evolutionary programming* and iv) *Genetic programming*. Each of these constitutes a different approach, however they are inspired by the same principle of natural evolution. They all share a common conceptual base of simulating the *evolution* of *individual* structures via processes of *selection*, *mutation*, and *crossover*. More precisely, EAs maintain a *population* of structures that evolve according to rules of *selection* and other operators, which are referred to as genetic operators, such as *crossover* and *mutation*. Let us have a discussion about all these evolutionary algorithms.

2.1. Genetic Algorithms

A g*enetic algorithm* developed by J.H. Holland (1975)[5][1] is a model of machine learning, which derives its behavior from a metaphor of the processes of e*volution* in nature. GAs are executed iteratively on a set of coded chromosomes, called a *population*, with three basic genetic operators: *selection*, *crossover* and *mutation*. Each member of the population is called a chromosome (or individual) and is represented by a string. GAs use only the objective function information and probabilistic transition rules for genetic operations. The primary operator of GAs is the c*rossover*. The figure 1shows the basic flow diagram of a GA:

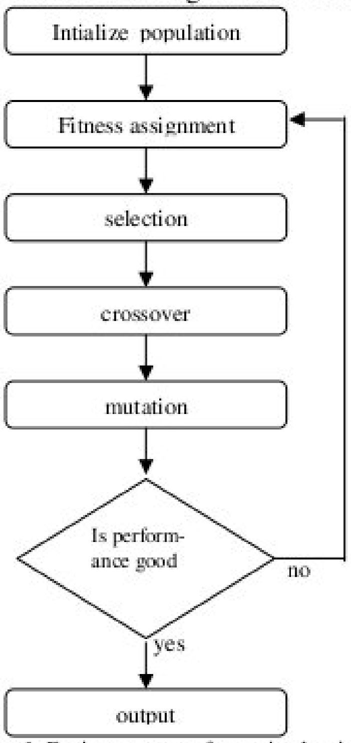

Figure 1. Basic structure of genetic algorithm

2.

Evolutionary strategies (ES) were developed in Germany by I.Rechenberg and H.P.Schwefel [6-8]. It imitates mutation, selection and recombination by using normally distributed mutation, a deterministic mechanism for selection (which chooses the best set of offspring individuals for the next generation) and a broad repertoire of recombination operators. The primary operator of the ES is mutation. There are two variants of selection commonly used in ES. In the elitist variant, the 'μ' parents individual for new generation are selected from the set of both 'μ' parents and 'λ' offspring at old generation. This is called *plus strategy* $(\mu+\lambda)$.

If parent individuals do not take part in the selection, then 'μ' individuals of the next generation are selected from 'λ' offspring, which is called *comma strategy* and denoted as (μ,λ). The notation $(\mu\ ,+\ \lambda)$ is used to subsume both selection schemes. Shows the basic principle of ES.

Algorithm

```
Begin
        t = 0
        initialize  $P_t$
        evaluate  $P_t$
    while (termination condition not satisfied) do
        begin
                t = t+1
                select $P_t$ from $P_{t-1}$
                mutate $P_t$
                evaluate $P_t$
        end
end
```

2.3 Evolutionary Programming

Fogel (1960) proposed a *stochastic optimization* strategy similar to GAs called evolutionary programming (EP)[9]. EP is also similar to evolutionary strategies (ESs). Like both ES and GAs, EP is a useful method of o*ptimization* when other techniques such as gradient descent or direct analytical discovery are not possible. Combinatorial and real-valued f*unction optimization,* in which the optimization surface or fitness landscape is "rugged", possessing many locally optimal solutions, are well suited for evolutionary programming. The basic EP method involves 3 steps (repeat until a certain number of iterations is exceeded or an acceptable solution is obtained):

- Generate the initial population of solutions randomly.
- Each solution is replicated into a new p*opulation*. Each of these *offspring* solutions are mutated according to a distribution of *mutation* types, ranging from minor to extreme with a continuum of mutation types. The severity of *mutation* is

judged on the basis of the functional change imposed on the p*arents*.

- Each *offspring* solution is assessed by computing it's *fitness*. Typically, a stochastic tournament is held to determine N solutions to be retained for the Population of solutions, although this is occasionally performed deterministically. There is no requirement that the Population size be held constant, however, not that only a single offspring is generated from each parent.

It should be pointed out that EP typically does not use any crossover as a genetic operator.

2.4 Genetic programming

Genetic programming (GP) was introduced by John Koza (1992)[10-12]. GP is the extension of the genetic model of learning into the space of programs. That is, the objects that constitute the p*opulation* are not fixed-length character strings that encode possible solutions to the problem at hand, they are programs that, when executed, are the candidate solutions to the problem. These programs are expressed in genetic programming as parse trees, rather than as lines of code. Thus, for example, the simple program $a + b * c$ would be represented as:

```
     +
    / \
   a   *
      / \
     b   c
```

or to be precise, as suitable data structures linked together to achieve this effect. Because this is a very simple thing to do in the programming language LISP, many genetic programmers tend to use LISP. There are straightforward methods to implement GP using a non-LISP programming environment also.

The programs in the Population are composed of elements from the function set and the terminal set, which are typically fixed sets of symbols selected to be appropriate to the solution of problems in the domain of interest.

In GP the crossover operation is implemented by taking randomly selected subtrees in the individual's (selected according to fitness) and exchanging them. It should be noted that GP usually does not use any mutation as in genetic operator.

3. Multi-objective evolutionary algorithms

Multi-objective optimization methods as the name suggests, deal with finding optimal solutions to problems having multiple objectives [13]. So in this type of problems the user is never satisfied by finding one solution that is optimum with respect to a single criterion.

The principle of a multi-criterion optimization procedure is different from that of a single criterion optimization. In a single criterion optimization the main goal is to find the global optimal solutions. However, in a multi-criterion optimization problem, there is more than one objective function, each of which may have a different individual optimal solution. If there is a sufficient difference in the optimal solutions corresponding to different objectives then we say that the objective functions are conflicting to each other. Multi-criterion optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution. The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all other objective functions. These optimal solutions have a special name called Pareto-optimal solutions following the name of an economist Vilfredo Pareto. He stated in 1896 a concept according to his name, known as "Pareto optimality".

The concept is that *the solution to a multi-objective optimization problem is normally not a single value but instead a set of values also called the "Pareto set"*. Let us illustrate the Pareto optimal solution with the time & space complexity of an algorithm shown in the following figure. In this problem we have to minimize both time as well as space complexity.

The point 'p' represents a solution, which has a minimal

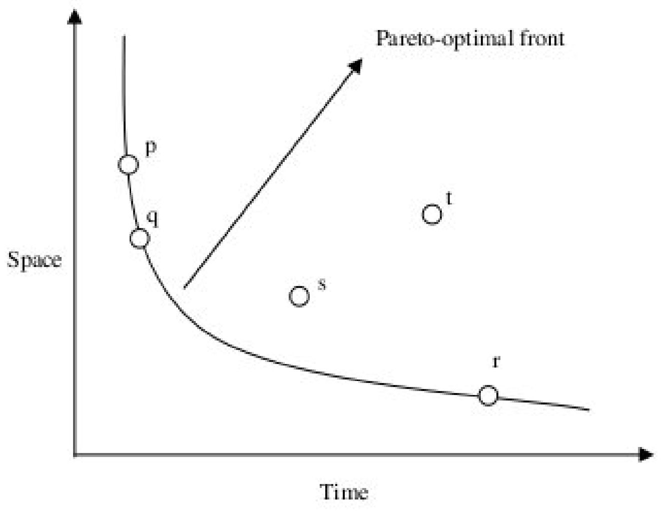

Figure 2. Pareto optimal solutions

time, but the space complexity is high. On the other hand, the point 'r' represents a solution with high time complexity but minimum space complexity. Considering both objectives, no solution is optimal. So in this case we can't say that solution 'p' is better than 'r'. In fact, there exist many such solutions like 'q' also belong to the *Pareto optimal set* and one can't sort the solution according to the performance metrics considering both objectives. All the solutions, on the curve, are known as *Pareto-optimal solutions*.

From the figure-2 it is clear that there exist solutions, which does not belong to the Pareto optimal set, such as a solution like 't'. The reason why 't' does not belong to Pareto optimal set is obvious. If we compare 't' with solution 'q', 't' is not better than 'q' with respect to any of the objectives. So we can say that 't' is a *dominated* solution or *inferior* solution.

Now we are clear about the optimality concept of multi-criterion. Based on the above discussions, we first define conditions for a solution to become dominated with respect to another solution and then present conditions for a set of solutions to become a Pareto-optimal set.

Let us consider a problem having m objectives (say $f_i, i = 1,2,3,....., m$ and $m > 1$). Any two solutions $u^{(1)}$ and $u^{(2)}$ (having 't' decision variables each) can have one of two possibilities-one dominates the other or none dominates the other. A solution $u^{(1)}$ is said to *dominate* the other solution $u^{(2)}$, if the following conditions are true:

1. The solution $u^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $u^{(2)}$ in all objectives, or $f_i(u^{(1)}) \geq f_i(u^{(2)}), \forall i = 1,2,3...., m$.
2. The solution $u^{(1)}$ is strictly better than $u^{(2)}$ in at least one objective, or $f_i(u^{(1)}) \succ f_i(u^{(2)})$ for at least one, $i \in \{1,2,3,.., m\}$.

If any of the above condition is violated, the solution $u^{(1)}$ does not dominate the solution $u^{(2)}$. If $u^{(1)}$ dominates the solution $u^{(2)}$, then we can also say that $u^{(2)}$ is dominated by $u^{(1)}$, or $u^{(1)}$ is non dominated by $u^{(2)}$, or simply between the two solutions, $u^{(1)}$ is the non-dominated solution.

The following definitions ensure whether a set of solutions belongs to a local or global Pareto-optimal set:

Local Pareto-optimal set

If for every member u in a set S, $\exists$ no solution v satisfying $\|u - v\|_\infty \leq \varepsilon$, where ε is a small positive number, which dominates any member in the set S, then the solutions belonging to the set S constitute a local Pareto-optimal set. The definition says that if we perturbing the solution u in a small amount then the resultant solution v dominates any member of that set then the set is called local Pareto optimal set

Global Pareto-optimal set

If there exits no solution in the search space which dominates any member in the set S, then the solutions belonging to the set S constitute a global Pareto-optimal set.

Difference between non-dominated set & a Pareto-optimal set

A *non-dominated set* is defined in the context of a sample of the search space (need not be the entire search space). In a sample of search points, solutions that are not dominated (according to the previous definition) by any other solution in the sample space constitute the non-dominated set. *A Pareto-optimal set* is a non-dominated set, when the sample is the entire search space. The location of the Pareto optimal set in the search space is sometimes loosely called as the Pareto optimal region.

From the above discussions, we observe that there are primarily two goals that a multi-criterion optimization algorithm must try to achieve:

1. *Guide the search towards the global Pareto-optimal region, and*
2. *Maintain population diversity in the Pareto-optimal front.*

The first task is a natural goal of any optimization algorithm. The second task is unique to multi-criterion optimization.

Multi-criterion optimization is not a new field of research and application in the context of classical optimization. The weighted sum approach, ε-perturbation method, Goal programming, Tchybeshev method, min-max method and others are all popular methods often used in practice [14]. The core of these algorithms, is a classical optimizer, which can at best, find a single optimal solution in one simulation. In solving multi-criterion optimization problems, they have to be used many times, hopefully finding a different Pareto-optimal solution each time. Moreover, these classical methods have difficulties with problems having non-convex search spaces.

Evolutionary algorithms (EAs) are a natural choice for solving multi-criterion optimization problems because of their population-based nature. A number of Pareto-optimal solutions can, in principle, be captured in an EA population, thereby allowing a user to find multiple Pareto-optimal solutions in one simulation run. Although the possibility of using EAs to solve multi-objective optimization problems was proposed in the seventies. David Schaffer first implemented Vector evaluated genetic algorithm (VEGA) in the year 1984. There was lukewarm interest for a decade, but the major popularity of the field began in 1993 following a suggestion by David Goldberg based on the use of the non-domination concept and a diversity- preserving mechanism. There are various multi-criterions EAs proposed so far, by different authors.

4. Different Approaches of MOEA

Following popular approaches have been used by different researchers for multi-objective optimization, each one having its merits and demerits.

- Weighted-sum-Approach (Using randomly generated weights and Elitism)
- Schaffer's Vector Evaluated Genetic Algorithm (VEGA)
- Fonseca and Fleming's Multi-Objective Genetic Algorithm (MOGA)
- Horn, Nafploitis, and Goldberg's Niched Pareto Genetic Algorithm (NPGA)
- Zitzler and Thiele's Strength Pareto Evolutionary Algorithm (SPEA)
- Srinivas and Deb's Non-dominated Sorting Genetic Algorithm (NSGA)
- Vector-optimized evolution strategy (VOES)
- Weight-based genetic algorithm (WBGA)
    - Sharing function approach
    - Vector evaluated approach
- Predator-prey evolution strategy (PPES)
- Rudolph's Elitist multi-objective evolutionary algorithm (REMOEA)
- Elitist non-dominated sorting genetic algorithm (ENSGA)
- Distance-Based Pareto genetic algorithm (DBPGA)
- Thermodynamical genetic algorithm (TGA)
- Pareto-archived evolution strategy (PAES)

4.1. Weighted-Sum Approach

This approach was proposed by Ishibuchi and Murata in 1996[15]. Here they used randomly generated weights and elitism. Let us see how this method works with the help of the following algorithm. Suppose there are m objective functions $f_i, i = 1,2,3,\dots,m$. Our task is to optimize (i.e. maximize) all these objective.

Algorithm

1. Generate the initial population randomly.
2. Compute the value of the m objectives for each individual in the population. Then determine which are the non-dominated solutions and store them in a separate population that we will call *external* to distinguish it from the current population, which we call *current*.
3. If n represents the number of non-dominated solutions in *external* and p is the size of *current*, then we select $p-n$ pairs of parents using the following procedure:
    - The fitness function used for each individual is $f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \dots w_m \cdot f_m(x)$ Generate m random number in the interval [0, 1] i.e. $r_1, r_2, \dots, r_m$. The weight values are determined as follow
    $$w_i = {r_i}\Big/{(r_1 + r_2 + \dots + r_m)}$$
    $\forall i = 1,2,3,\dots,m$. This ensure that all $w_i \geq 0$, $i = 1,2,3\dots,m$ and that $w_1, w_2, \dots w_m = 1$.
    - Select a parent with probability $p(x) = \dfrac{f(x) - f_{\min}(current)}{\sum_{x \in current}\left(f(x) - f_{\min}(current)\right)}$, where $f_{\min}$ is the minimum fitness in the current population.
4. Apply crossover to the selected $p-n$ pairs of parents. Apply mutation to the new solutions generated by crossover.
5. Randomly select n solutions from *external*. Then add the selected solutions to the $p-n$ solutions generated in the previous step to construct a population of size p.
6. Stop if a pre-specified stopping criterion is satisfied (e.g., the pre-specified maximum number of generations has been reached). Otherwise, return to step 2.

Merits & Demerits

The advantage of this method is, it's computational efficiency, and its suitability to generate a strongly non-dominated solution that can be used as an initial solution for other techniques.

Its main demerit is the difficulty to determine the weights that can appropriately scale the objectives when we do not have enough information about the problem, particularly if we consider that any optimal point obtained will be a function of such weights. Still more important is the fact that this approach does not generate proper Pareto-optimal solutions in the presence of non-convex search spaces regardless of the weights used. However the incorporation of elitism is an important aspect of this method.

4.2. Schaffer's Vector Evaluated Genetic Algorithm (VEGA)

David Schaffer (1984)[16-18] extended Grefenstette's GENESIS program [Grefenstette 1984] to include multiple objective functions. Schaffer's approach was to use an extension of the *simple genetic algorithm* (SGA) that he called the *vector evaluated genetic algorithm* (VEGA), and that differed from the SGA in selection only. This operator was modified so that at each generation a number of sub-populations were generated by performing proportional selection according to each objective in turn. Thus for a problem with 'k' objectives, 'k' sub-populations of size N/k would be generated (assuming a total population size of N). These sub-populations would be shuffled together to obtain a new population of size N, on which the GA would apply, the crossover and mutation operators in the usual way. Figure 3 shows the structural representation of this process. Schaffer realized that the solutions generated by his system were non-dominated in a local sense, because their non-dominance was limited to the current population, and while a local dominated individual is also globally dominated, the converse is not true [Schaffer 1985]. An individual who is dominated in one generation may become dominated by an individual who emerges in a later generation. Also he noted a problem that in genetics is known as '*speciation*' (i.e. we could have the evolution of 'species' within the population which excel on different aspects of performance). This problem arises because this technique selects individuals who excel in one dimension of performance, without looking at the other dimensions. The potential danger doing it is that we could have individuals with what Schaffer calls "middling" performance in all dimensions, which could be very useful for compromise solutions, but that will not survive under this selection scheme, since they are not in the extreme for any dimension of performance (i.e. they do not produce the best value for any objective function, but only moderately good values for all of them). Speciation is undesirable because it is against our goal of finding a compromise solution. Schaffer tried to minimize this speciation by developing two heuristics – the *non-dominated selection heuristic* (a wealth redistribution scheme), and the *mate selection heuristic*(a cross breeding scheme).

In the non-dominated selection heuristic, dominated individuals are penalized by subtracting a small fixed penalty from their expected number of copies during selection. Then the total penalty for dominated individuals was divided among the non-dominated individuals and was added to their expected number of copies during selection.

But this algorithm failed when the population has very few non-dominated individuals, resulting in a large fitness value for those few non-dominated points, eventually leading to a high selection pressure. The mate selection heuristic was implemented by the following procedure:

Initially select an individual randomly from the population. Then the corresponding mate was selected as the individual, which has maximum Euclidean distance from it.

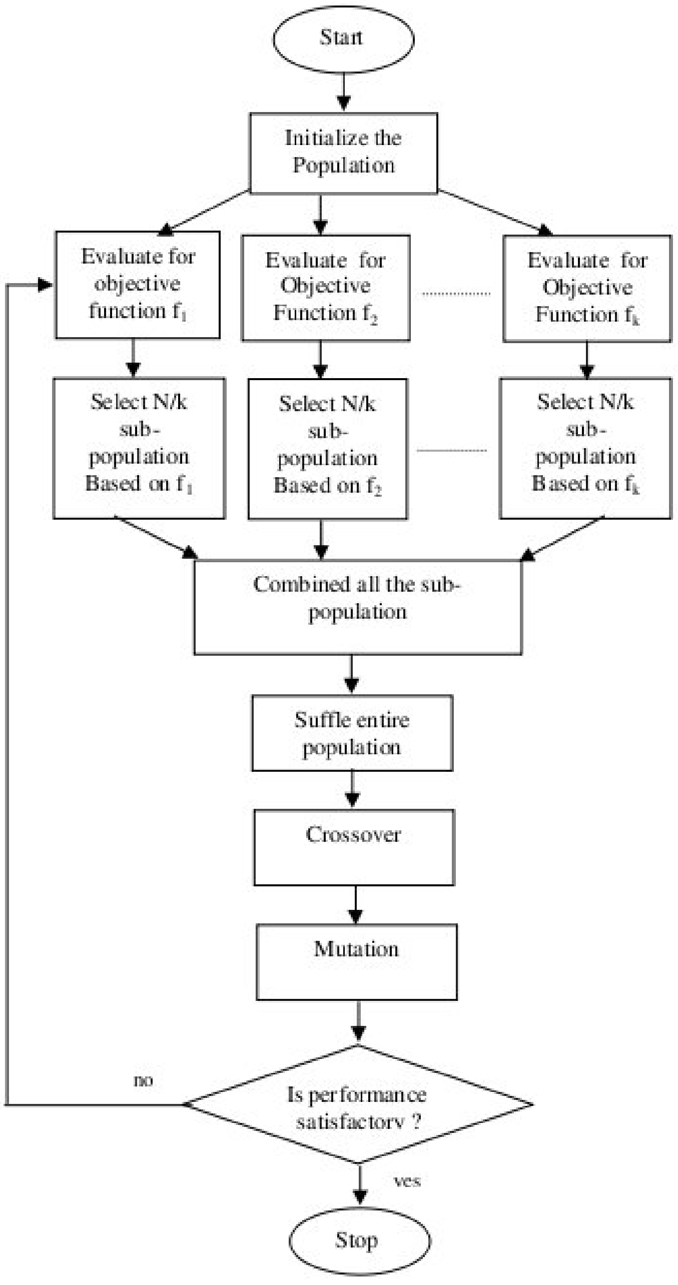
Start
Initialize the Population
Evaluate for objective function f_1
Evaluate for Objective Function f_2
Evaluate for Objective Function f_k
Select N/k sub-population Based on f_1
Select N/k sub-population Based on f_2
Select N/k sub-population Based on f_k
Combined all the sub-population
Suffle entire population
Crossover
Mutation
no
Is performance satisfactorv ?
ves
Stop

Figure 3 Schematic Representation of VEGA

But it failed too to prevent the participation of poorer individuals in the mate selection. This is because of random selection of the first mate and the possibility of a large Euclidean distance between a champion and a less fitted solution. Schafer concluded that the random mate selection is far superior than this heuristic.

Merits & Demerits

The main advantage of this algorithm is its simplicity i.e. this approach is easy enough to implement. Richardson et al.(1989)[19] noted that the shuffling and merging of all the sub-populations corresponds to averaging the fitness components associated with each of the objectives. Since Schaffer used proportional fitness assignment, these fitness components were in turn proportional to the objectives themselves. Therefore, the resulting expected fitness corresponds to a linear combination of the objectives where the weights depend on the distribution of the population at each generation as shown by Richardson et al.

The main consequence of this is that when we have a concave trade-off surface certain points in concave regions will not be found through this optimization procedure in which we are using just a linear combination of the objectives, and it has been proved that this is true regardless of the set of weights used. Therefore, the main weakness of this approach is its inability to produce Pareto-Optimal solutions in presence of non-convex search spaces.

4.3. Fonseca and Fleming's MOGA

Fonseca and Fleming (1993)[20-22] implemented Goldberg's suggestion in a different way. Let us first discuss what is Goldberg's suggestion about Pareto ranking.

Goldberg proposed the Pareto ranking scheme in (1989), where a solution 'x' at generation 't' has a corresponding objective vector x_u, and 'n' is the population size, the solution's rank is defined by the following algorithm.

Algorithm

```
curr_rank = 1
m = n
While n != 0 do
        For i= 1. . . ,m
        do
         If $x_u$ is non-dominated
        Rank (x, t) = curr_rank
        enddo
        For i=1,.....,m
         do
        If rank (x, t)= curr_rank
        Remove x from population
        n= n-1;
        enddo
        curr_rank = curr_rank +1
        m = n
endwhile
```

This means, in Pareto ranking, the whole population is checked and all non-dominated individuals are assigned rank '1'. Then remove these individuals from the population, which has rank '1'. Again detect all non-dominated individual from the rest of the population and assign rank 2. In this way the procedure goes on till all solutions have been assigned the required rank.
But in MOGA, the whole population is checked and all non-dominated individuals are assigned rank '1'. Other individuals are ranked by checking the non-dominance of them with respect to the rest of the population in the following way.

For example, an individual x_i at generation t, which is dominated by $p_i^{(t)}$ individuals in the current generation. Its current position in the individual's rank can be given by

$$\text{Rank}(x_i, t) = 1 + p_i^{(t)}$$

After completing the ranking procedure, then it is time to assign the fitness to each individual. Fonseca and Fleming proposed two method of assigning fitness.

- Rank-based fitness assignment method
- Niche-formation methods.

Rank-based fitness assignment is performed in the following way [1993]

1. Sort population according to rank
2. Assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank $n \leq N$) in the usual way, according to some function, usually linear but not necessarily.
3. Average the fitness of individuals with the same rank, so that all of them will be sampled at the same rate. This procedure keeps the global population fitness constant while maintaining appropriate selection pressure, as defined by the function used.

As Goldberg and Deb have pointed out, this type of blocked fitness assignment is likely to produce a large selection pressure that might produce premature convergence. To avoid this, Fonseca and Fleming use the second method (i.e. *Niche-formation method*) to distribute the population over the Pareto optimal region, but instead of performing sharing on the parameter values, they have used sharing on the objective function values.

Merits & Demerits

The main advantage of MOGA is that it is efficient and relatively easy to implement. The performance of this method is highly dependent on the sharing factor (σ_{share}). However Fonseca and Fleming have developed a good methodology to compute such a value for their approach.

4.4. Horn, Nafploitis, and Goldberg's NPGA

Horn, Nafploitis, and Goldberg [23][24] proposed the NPGA based on *Pareto domination tournament* and *equivalence class sharing*.

Pareto domination tournament

This is basically a tournament selection scheme based on the Pareto dominance. In this selection scheme, a comparison set comprising of a specific number (t_{dom}) of individuals is picked at random from the population at the beginning of each selection process. Two candidates for selection are picked at random from the population. Each of the candidates is then compared against each individual in the comparison set. If one is dominated by a comparison set and the other is not, the later is selected for reproduction. If neither or both are dominated by the comparison set, then we proceed to the second technique.

Equivalence class sharing

Since both individuals are same i.e. either dominated or non-dominated, it is likely that they are in the same equivalence class. So in this case we choose the "best fit" by the following procedure.

We choose the niche radius (σ_{share}) and according to that radius, candidates which have the least number of individuals in the population are the "best fit". The following figure-4 shows how this procedure work: here we are maximizing along x-axis and minimizing on the y-axis.

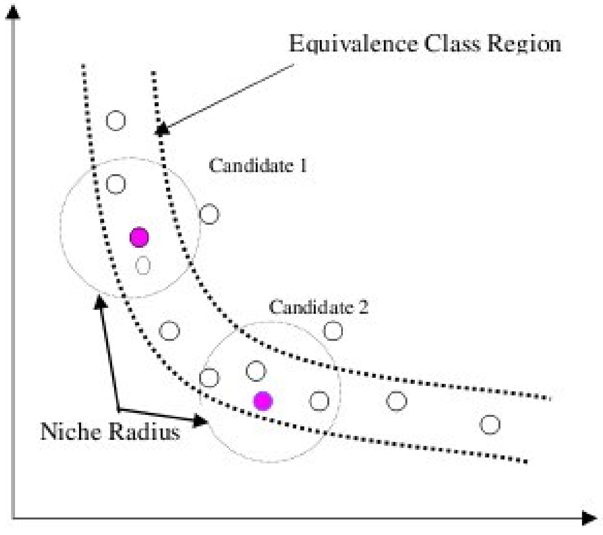

Fig. 3 Equivalence Class Sharing

In this case the two candidates for selection are not dominated by the comparison set. So by niche count this shows that candidate 1 is best fit.

Here t_{dom} is chosen only once for a particular generation 't'. After generating the new population apply the genetic operator similar to other methods.

Algorithm

1. Initialize the population size of n.
2. Choose a t_{dom} size of chromosome randomly from the current population.
3. Randomly pick up two chromosomes from the current population.
4. Select n individuals based on the following procedure: Compare the two chromosome with t_{dom} for non-domination by the former definition. If one is dominated and other is non-dominated then select the non-domination one. If both are non-dominated or dominated then by niche formation method then choose the best one.
5. Apply crossover and mutation to give the new population.
6. Test whether the performance criteria is satisfied or not, if not then goto step 2 otherwise step 7.
7. Stop.

Merits & Demerits

Since this approach does not apply Pareto selection to the entire population, but only to a segment of it at each run, its main advantages are that is very fast and that it produces good non-dominated fronts that can be kept for a large number of generations.

The performance of this method is highly dependent on sharing factor (σ_{sahre}) and the good choice of tournament size (t_{dom}) and complicated to implement.

4.5. Zitzler and Thiele's Strength Pareto Approach (SPEA)

Zitzler and Thiele's[25-27] suggested this approach with the combination of elitism and concept of non-domination. The following procedure gives the details:

At every generation they are maintaining an external population called EXTERNAL-storing a set of non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations.

The fitness of each individual in the current population and in the external population is decided based on the number of dominated solutions. Specifically the following procedure is adopted. First of all combine the external population and current population. Then assign the fitness to all the non-dominated solutions based on the number of solutions they dominate. Generally the highest fitness is assigned to the non-dominated solutions having more dominated solution in the combined population. After generating a population for next generation, it is required to update the external population. The updating procedure is as follows:

For each individual in the current population, check whether it is dominated by, both current population and external population. If it is not dominated by both then add this individual to external population (EXTERNAL). Then all solutions dominated by the added one are eliminated from external population. In this way the external population is updated in every generation. The following algorithm shows the idea:

Algorithm

1. Initialize the population of size n.
2. Determine all the non-dominated solutions from current population and store them in a separate population called EXTERNAL.
3. Combine the external with current population and assign the fitness to individuals according to how many solutions it dominates.
4. After assigning the fitness to all the individuals, select n individuals for next generation.
5. Apply crossover and mutation to get the new population of size n.
6. Update the EXTERNAL population.
7. Test whether the performance criterion is satisfied or not. If not then goto step 3.Otherwise stop.

Zitzler and Thieles[25] solved the 0/1-Knapsack problem using this approach. They have reported better results than other methods.

Merits & Demerits

The main merits of this method are that, it shows the utility of introducing elitism in evolutionary multi-criterion optimization. Since this method does not use sharing factor σ_{share} & t_{dom} as in the MOGA and NPGA; this facilitates good advantages.

There exists a multi-objective problem where the resulting Pareto-optimal front is non-convex. In that case GA's success to maintain diverse Pareto-optimal solutions largely depends on fitness assignment procedure. This method does not converge to true Pareto-optimal solutions, because this method uses the fitness assignment procedure, which is very sensitive to concave surface.

4.6. Srinivas and Deb's Non-dominated Sorting Genetic Algorithm (NSGA)

Non-dominated GAs varys from simple GAs only in the way the selection operator in used. The crossover and mutation operators remain as usual [28-30]. Let us discuss the selection operator they proposed.

For selection, two steps are needed. First, the population is ranked on the basis of an individual's non-domination level and then sharing is used to assign fitness to each individual.

Ranking individuals based on non–domination level

Consider a population of size n, each having m (>1) objectives function values. The following algorithm can be used to find the non-dominated set of solutions:

Algorithm

```
For i = 1  to  n  do
   For j = 1  to  n  do
      If  (j != i) then
         Compare solutions $x^{(i)}$ and
         $x^{(j)}$ for domination using two
         conditions for all m objectives.
         if for any j , $x^{(i)}$ is  dominated by $x^{(j)}$ ,
         mark  $x^{(i)}$  as dominated
      endif
   endfor
endfor
```

Solutions, which are not marked "dominated", are non-dominated solution. All these non-dominated solutions are assumed to constitute the first non-dominated front in the population. In order to find the solutions belonging to the second level of non-domination, we temporarily disregard the solutions of the first level of non-domination and follow the above procedure. The resulting non-dominated solutions are the solutions of the second level of non-domination. This procedure is continued till all solutions are classified into a level of non-domination. It is important to realize that the number of different non-domination levels could vary between 1 to n.

Fitness assignment

In this approach the fitness is assigned to each individual according to its non-domination level. An individual in a higher level gets lower fitness. This is done in order to maintain a selection pressure for choosing solutions from the lower levels of non-domination. Since solutions in lower levels of non-domination are better, a selection mechanism that selects individuals with higher fitness provides a search direction towards the Pareto-optimal region.

Setting a search direction towards the Pareto-optimal region is one of the two tasks of multi-objective optimization. Providing diversity among current non-dominated solutions is also important in order to get a good distribution of solutions in the Pareto-optimal front. The fitness assignment is performed in two stages.

1. Assigning same dummy fitness to all the solutions of a particular non-domination level.
2. Apply the sharing strategy.

Now we discuss the details of these two stages:

First, all solutions in the first non-dominated front are assigned a fitness equal to the population size. This becomes the maximum fitness that any solution can have in any population. Based on the sharing strategy, if a solution has many neighboring solutions in the same front, its dummy fitness is reduced by a factor and a shared fitness is computed. The factor depends on the number and proximity of neighboring solutions. Once all solutions in the first front are assigned their fitness values, the smallest shared fitness value is determined.

Thereafter, the individuals in the second non-domination level are all assigned a dummy fitness equal to a number smaller than the smallest shared fitness of the previous front. This makes sure that no solution in the second front has a shared fitness better than that of any solution in the first front. This maintains a pressure for the solutions to lead towards the Pareto-optimal region. The sharing method is again used among the individuals of second front and shared fitness of each individual is found. This procedure is continued till all individuals are assigned a shared fitness.

After the fitness assignment method, use a stochastic remainder roulette-wheel selection for selecting N individuals. Thereafter apply the crossover and mutation. Shared fitness is calculated as follows:

Given a set of n_1 solutions in the l^th^ non-dominated front each having a dummy fitness value f_1, the sharing procedure is performed in the following way for each solution $i = 1,2,3,\ldots,n_1$.

1. Compute a normalized Euclidean distance measure with another solution j in the l^th^ non-dominated front, as follows:

$$d_{ij} = \sqrt{\sum_{p=1}^{P}\left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^{(u)} - x_p^{(s)}}\right)^2}$$

where 'P' is the number of variables in the problem. The parameter's $x_p^{(u)}$ and $x_p^{(s)}$ are the upper and lower bounds of variable x_p.

2. This distance d_{ij} is compared with a pre-specified parameter σ_{share} and the following sharing function value is computed:

$$sh(d_{ij}) = \begin{cases} 1 - \left(d_{ij}/\sigma_{share}\right)^2, & if d_{ij} \le \sigma_{share} \\ 0, & otherwise \end{cases}$$

3. Increment j. If $j <= n_1$, go to step 1 and calculate $sh(d_{ij})$. If $j \succ n_1$, calculate niche count for i^th^ solution as follows: $m_i = \sum_{j=1}^{n_1} sh(d_{ij})$

(iv). Degrade the dummy fitness f_1 of i^th^ solution in the l^th^ non-domination front to calculate the shared fitness, $f_i^{'}$, as follows: $f_i^{'} = \frac{f_1}{m_i}$

This procedure is continued for all $i = 1,2,3\ldots,n_1$ and a corresponding $f_i^{'}$ is found. Thereafter, the smallest value $f_1^{\min}$ of all in $f_i^{'}$ the l^th^ non-dominated front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{l+1} = f_1^{\min} - \varepsilon_1$, where ε_1 is a small positive number.

The above sharing procedure requires a pre-specified parameter σ_{share}, which can be calculated as follows:

$$\sigma_{share} \approx 0.5 / \sqrt{q}^{\,1/p},$$

where q is the desired number of distinct Pareto-optimal solutions. Although the calculation of σ_{share} depends on this parameter q, it has been shown that $q \approx 10$ works well for many test problems.

Merits & Demerits

The main advantage of this method is that it can handle any number of objectives, and that does sharing in the parameter value space instead of the objective value space, which ensures a better distribution of individuals, and allows multiple equivalent solutions.

Some researcher's point out that it is inefficient than MOGA, if we consider both computational efficiency of the Pareto fronts produced. Another disadvantage is that it is more sensitive to σ_{share}.

4.7. Vector-Optimized Evolution Strategy (VOES)

VOES was suggested by Frank Kursawe (1990)[31] for multi-objective optimization. Here the fitness assignment scheme is similar to the VEGA, but Kursawe's algorithm used a number of other aspects from nature.

Using a diploid chromosome, each having a dormant and a recessive string represents the solution. Two different solution vectors (each with a decision variable vector $\vec{x}$ and the corresponding strategy parameter vector σ) are used as an individual in a population. Thus an individual i results in two objective vector evaluations: (i) f_d, calculated by using the decision variable of the dominant genotype and (ii) f_r, calculated by using the decision variables of the recessive genotype. Let us see the procedure of fitness evaluation and selection scheme:

Let n be the number of objective functions. Then the selection process is completed in n steps. For each step a user defined probability is used to choose an objective. This vector can either be kept fixed or varied with generations. If the mth objective function is chosen, the fitness of an individual i s computed as a weighted average of its dominant objective value $(f_d)_m^i$ and recessive objective value $(f_r)_m^i$ in a 2 : 1ratio.

$$F^i = (2/3)(f_d)_m^i + (1/3)(f_r)_m^i$$

For each selection step, the population is sorted according to each objective function and the best $((n-1)/n)^{th}$ portion of the population is chosen as parents. This procedure is repeated 'n' times, every time using the survived population from the previous sorting. Thus the relationship between λ and μ is as follows:

$$\mu = ((m-1)/m)^m \cdot \lambda$$

All new μ solutions are copied into an external set. Which stores all non-dominated solutions found since the beginning of the simulation run. After new solutions are inserted into the external set, a non-domination check is made on the whole external set and only new non-dominated solutions are retained. If the size of the external set exceeds a certain limit, a niching mechanism is used to eliminate closely packed solutions. This allows maintenance of diversity in the solutions of the external set.

Merits & Demerits

We observe that this algorithm performs a domination check to retain non-dominated solutions and a niching mechanism to eliminate crowded solutions. These features are essential in a good MOEA. Since Kursawe has not simulated it for multi-objective optimization, it remains a challenging task to investigate how this algorithm will fare in more complex problems. Because of its complications it is not much used by current researchers.

4.8 Weight–based Genetic Algorithm (WBGA)

Hajela and Lin (1993)[32[33] proposed a weight–based genetic algorithm for multi-objective optimization. As the name suggests, each objective function f_i is multiplied by a weight w_i. A GA string represents all decision variables x_i as well as their associated weights w_i. The weighted objective function values are then added together to calculate the fitness of a solution. Each individual in a GA population is assigned a different weight vector. The key issue in WBGAs is to maintain diversity in the weight vectors among the population members. In WBGAs the diversity in the weight vectors is maintained in two ways:

- A niching method is used only on the sub-string representing the weight vectors.
- Carefully chosen subpopulations are evaluated for different predefined weight vectors, an approach similar to that of VEGA.

Sharing function approach

In addition to n decision variables the user codes a weight vector of size M in a string, the weights must be chosen in a way so as to make the sum of all weights in every string one.

$$w_i = w_i / [\sum w_j]$$

Otherwise if only a few Pareto-optimal solutions are desired, a fixed set of weight vectors may be chosen before hand and an integer variable x_w can be used to represent one of the weight vectors, w^k.

A solution $x^{(i)}$ is then assigned a fitness equal to the weighted sum of the normalized objective function values computed as follows:

$$F(x^{(i)}) = \sum w_{j_w}^{(x(i))} \left((f_j x(i) - f_j^{\min}) / (f_j^{\max} - f_j^{\min}) \right)$$

In order to preserve diversity in the weight vectors, a sharing strategy is proposed:

$$d_{ij} = \left| x_w^{(i)} - x_w^{(j)} \right|$$

The sharing function $sh(d_{ij})$ is computed with a niching parameter σ_{share}. Next calculate the shared fitness F'.

Algorithm

1. For each objective function j, set upper and lower bounds as $f_j^{\max}$ and $f_j^{\min}$.
2. For each solution $i = 1,2,3,...,n.$ calculate the distance

$$d_{ik} = \left| x_w^{(i)} - x_w^{(k)} \right|$$

with all solutions $k = 1,2,3...,n$. Then calculate the sharing function values as follows:

$$sh(d_{ik}) = \begin{cases} d_{ik}/\sigma_{share}, if d_{ik} \le \sigma_{share} \\ 0, otherwise \end{cases}$$

Thereafter, calculate the niche count of the solution i as $n_{ci} = \sum sh(d_{ik})$.

3 For each solution $i = 1,2,....,n$ follow the entire procedure below.
Assign fitness F_i according to $F(x^i)$ calculate the shared fitness as $F_i^{'} = F_i / n_{ci}$.

When all population members are assigned fitness F', the proportionate selection is applied to create the mating pool. Thereafter crossover and mutation operators are applied on the entire string, including the sub-string representing x_w.

Merits & Demerits

Since a WBGA uses a single–objective GA, not much change is needed to convert a simple GA implementation into a WBGA one. Moreover, the complexity of the algorithm (with a single extra variable x_w) is smaller than other multi-objective evolutionary algorithms.

The WBGA uses a proportionate selection procedure on the shared fitness values. Thus, in principle, the WBGA will work in a straightforward manner in finding solutions for maximization problems. However, if objective functions are to be minimized, they are required to be converted into a maximization problem. Moreover, for mixed types of objective functions (some are to be minimized and some are to maximized), complications may arise in trying to construct a fitness function.

Weighted sum approaches share a common difficulty in finding Pareto-optimal solutions in problems having non-convex Pareto-optimal region. Thus, a WBGA may also face difficulty in solving such problems.

Vector evaluated approach

This approach is similar to VEGA. First a set of K different weight vectors w_k (where k= 1,2,.. K) are chosen. Thereafter, each weight vector w_k is used to compute the weighted normalized fitness for all n population members. Among them the best n/K members are grouped together into a subpopulation. Perform selection, crossover and mutation on p_k to create a new population of size n/K, if k < K, increment k by one and go to step-2. Otherwise, combine all subpopulations to create the new population $p = \cup p_k$. If |p| < n, add randomly created solutions to make the population size equal to n.

Merits & Demerits

As in any weight-based method, knowledge of the weight vectors is also essential in this approach. This method is better in complexity that the sharing function approach, because no pair-wise distance metric is required here. There is also no need for keeping any additional variable for the weight vectors. However, since GA operators are applied to each subpopulation independently, an adequate subpopulation size is required for finding the optimal solution corresponding to each weight vector. This requires a large population size.

By applying both methods in a number of problems, investigators concluded that the vector-evaluated approach is better that the sharing function approach.

4.9. Predator-prey evolution strategy (PPES)

Laumanns et al. (1998)[34] suggested a multi-objective evolution strategy (ES), which is different from any of the methods discussed so far. 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=1,2,3.. ,n) which are placed on the vertices of a undirected connected graph. First each predator is randomly placed on any vertex of the graph. Each predator is associated with a particular objective function. Thus at least M predators are initialized in the graph. Secondly, staying in 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 in the vertex. After this event is over, the predator takes a random walk to any of its neighboring vertices. The above procedure continues in parallel (or sequential) with all predators until a pre-specified number of iterations have elapsed.

Merits & Demerits

The main advantage of this method is its simplicity. Random walks and replacing worst solutions of individual objectives with mutated solutions are all simple operations, which are easy to implement. Another advantage of this algorithm is that it does not emphasize non-dominated solutions directly. Experimentally shown that the algorithm is sensitive to the choice of mutation strength parameter and the ratio of predator and prey. No explicit operator is used to maintain a spread of solutions in the obtained non-dominated set.

4.10. Thermodyanamical genetic algorithm (TDGM)

Kita et al. (1996) [35] suggested a fitness function, which allows a convergence near the Pareto-optimal front and a diversity-preservation among obtained solutions. The fitness function is motivated from the thermodynamic equilibrium condition, which corresponds to the minimum of the Gibb's energy F, defined as follows:

$$F = \langle E \rangle - HT \,,$$

where $\langle E \rangle$ is the mean energy, H is the entropy, and T is the temperature of the system. The relevance of the above term in multi-objective optimization is as follows. The mean energy is considered as the fitness measure. The second term HT controls the diversity existent in the population members. In this way minimization of F means minimization of $\langle E \rangle$ and maximization of HT. In the parlance of multi-objective optimization, this means minimization of the objective function (convergence towards the Pareto-optimal set) and the maximization of diversity in obtained solutions. Since these two goals are precisely the focus of an MOEA, the analogy between the principle of finding the minimum free energy state in a thermodynamic system and the working principle of an MOEA is applicable.

Algorithm

1. Select a population size N, maximum number of generations $t_{\max}$ and an annealing schedule $T(t)$ (monotonically non-increasing function) for the temperature variation.
2. Set the generation counter t=0 and initialize a random population $P(t)$ of size N.
3. Using crossover & mutation operator, create the offspring population $Q(t)$ of size N from $P(t)$. Combine the parent and offspring populations together and call this

   $$R(t) = P(t) \cup Q(t) \,.$$

4. Create an empty set $P(t+1)$ and set the individual counter i of $P(t+1)$ to one.
5. For every solution $j \in R(t)$, construct $P'(t, j) = P(t+1) \cup \{j\}$ and calculate the free energy fitness of the j^{th} member as follows: $F(j) = \langle E(j) \rangle - T(t) \sum H_k(j)$ where $\langle E(j) \rangle \geq \sum E_k / |P'(t, j)|$ and the entropy is defined as $H_k(j) = -\sum P_1^k \log P_1^k$. The term P_1^k is the proportion of bit '1' on the locus k of the population $P'(t, j)$.

   After all 2N population members are assigned a fitness, find the solution $j* \in R(t)$ which will minimize F given above . Include $j* \in R(t+1)$.
6. If $i \prec N$, then $i = i+1$ and go to step 5.
7. If $t \prec t_{\max}$, then $t = t+1$ and go to step 3.

The term E_k can be used as the non-dominated rank of the k^th solution in the population $P'(t, j)$.

Merits & Demerits

Since both parents and offspring populations are combined and the best set of N solutions are selected from the diversity preservation point of view, the algorithm is an elitist algorithm.

The TDGA requires a predefined annealing schedule T(t)- It is interesting to note that this temperature parameter acts like a normalization parameter for the entropy term need to balance between the mean energy minimization and entropy maximization. Since this requires a crucial balancing act for finding a good convergence and spread of solutions, an arbitrary T(t) distribution may not work well in most problems.

4.11. Pareto-archived evolution strategy (PAES)

Knowles and Corne (2000) suggested a MOEA, which uses an evolution strategy. In its simplest form, the PAES uses a (1+1)- ES. The main motivation for using an ES came from their experience in solving real world telecommunications network design problem.

At first, a random solution x_0 (call this a parent) is chosen. Using a normally distributed probability function with zero mean and with fixed mutation strength then mutates it. Let us say that the mutated offspring is C_0. Now both of these solutions are compared and the winner becomes the parent of the next generation. The main crux of the PAES lies in the way that a winner is chosen in the midst of multiple-objectives.

At any generation t, in addition to the parent P_t and the offspring C_t, the PAES, maintains an archive of the best solutions found so far. Initially, this archive is empty and as the generations proceed, good solutions are added to the archive and updated. However a maximum size is always maintained. First the parent P_t and the offspring C_t are compared for domination. It generates three scenarios:

- If P_t dominates C_t the offspring C_t is not accepted and a new mutated solution is created for further processing.
- If C_t dominates P_t, the offspring is better than the parent. Then solution C_t is accepted as a parent of the next generation and as copy of it is kept in the archive.
- If both P_t and C_t are non-dominated to each other, then the offspring is compared with the current archive:

- The offspring is dominated by a number of the archive. This means this is not a worth candidate, so the parent P_t again mutated to find new offspring for further processing.
- The offspring dominates a member of the archive. Then the dominated members are deleted and the offspring is accepted. The offspring becomes the parent of the next generation.
- The offspring is not dominated by any member of the archive and it also does not dominate any member of the archive. In this case, it belongs to that non-dominated front in which the archive solution belongs. In such a case the offspring is added to the archive only if there is a slot available in the latter.

The acceptance of the offspring in the archive does not qualify it to become the parent of the next generation. This is because as for the offspring C_t, the current parent P_t is also a member of the archive.

To decide who qualifies as a parent of the next generation, the density of solution in their neighborhood is checked. The one residing in the least crowded area in the search space qualifies as the parent. If the archive is full, the above density-based comparison is performed between the parent and the offspring to decide who remains in the archive. The density calculation is different from the other methods we have discussed so far.

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 hypercubes. The archived solutions are placed in these hypercubes according to their locations in the objective space. There after the number of solutions in each hypercube is counted. If the offspring resides in a less crowded hypercube 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.

Algorithm (Single iteration)

Parent, $P_{t,}$ offspring C_t and an archive A_t. The archive is initialized with the initial random parent solution.

1. If C_t is dominated by any member of A_t, set $P_{t+1} = P_t$ (A_t is not updated). Process is complete. Otherwise, if C_t dominates a set of members from A_t:
   $D(C_t) = \{i \mid i \in A_t, C_t \le i\}$, perform the following step.
   $$A_t = A_t / D(C_t)$$
   $$A_t = A_t \cup \{C_t\}$$
   $$P_{t+1} = C_t$$
   Process is complete. Otherwise go to step 2.

2. Count the number of archived solutions in each hypercube. The parent P_t belongs to a hypercube having n_p solutions. While the offspring belongs to a hypercube having n_c solutions. The highest count hypercube contains the maximum number of archived solutions.
   If $|A_t| \prec N$, include the offspring in the archive or $A_t = A_t \cup \{C_t\}$and $P_{t+1} = winner(C_t, P_t)$ and return. Otherwise (if $|A_t| = N$), check if C_t belongs to the highest count hypercube. If yes reject C_t, set $P_{t+1} = P_t$, and return. Otherwise replace a random solution r from the highest count hypercube with C_t:
   $$A_t = A_t / \{r\}$$
   $$A_t = A_t \cup \{C_t\}$$
   $$P_{t+1} = winner(C_t, P_t)$$
   The process is complete. The winner (C_t, P_t)chooses C_t, if $n_c < n_p$.
   Otherwise it chooses P_t.

Merits & Demerits

The PAES has a direct control on the diversity that can be archived in the Pareto –optimal solutions step-2 of the algorithm emphasizes the less populated the hypercubes to survive, thereby ensuring diversity. Choosing an appropriate value of d can control the size of these hypercubes. In addition to choosing an appropriate archive size (n), the depth parameter d, which directly controls the hypercube size, is also an important parameter.

4.12. Rudolph's Elitist Multi-objective Evolutionary Algorithm (REMOEA)

Rudolph (2001)[36][37] suggested, but did not simulate, a multi-objective evolutionary algorithm that uses elitism. Nevertheless, the algorithm provides a useful plan for introducing elitism into a multi-objective EA. In its general format, parents are used to create 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 steps as follows:

Algorithm

1. Create the parent population P_t to create an offspring population Q_t. Find non-dominated solutions of Q_t and put them in Q^*. Delete Q^* from Q_t or $Q_t = Q_t \setminus Q^*$. Set $P' = Q' = 0$.
2. For each $q \in Q^*$, find all solutions $p \in P_t$ that are dominated by q, put these solutions of P_t in a set $D(q)$. If $|D(q)| \succ 0$, carry out the following:
$$P_t = P_t / D(q)$$
$$P' = P' \cup \{q\}$$
$$Q^* = Q^* / \{q\}$$
Otherwise, if $D(q) = 0$ and q is non-dominated with all existing members of $P(t)$, then perform the following:
$$Q' = Q' \cup \{q\},\ Q^* = Q^* / \{q\}$$
3. Form a combined population $P_{t+1} = P_t \cup P'$. If $|P_{t+1}| \prec \mu$, then fill P_{t+1} with elements from the following sets in the order mentioned below until $|P(t+1)| = (Q', Q^*, Q_t)$.

Merits & Demerits

Rudolphs has proven that the above algorithm with a 'positive variation kernel' of its search operators allows convergence (of at least one solution in the population) to the Pareto-optimal front in a finite number of trials in finite search space problem. However, what was not been proved is a guaranteed maintenance of diversity among the Pareto–optimal solutions. If a population–based evolutionary algorithm converges to a single Pareto-optimal solution, it cannot be justified for its use over other classical algorithms. What makes evolutionary algorithms attractive is their ability to find multiple Pareto-optimal solutions in one single run.

4.13. Elitist Non-dominated Sorting Genetic Algorithm (ENSGA)

Like Rudolph's algorithm, in ENSGA, the offspring population Q_t is first created by using the parent population P_t. However, instead of finding the non-dominated front of Q_t, only, first the two populations are combined together to form R_t of size 2N. Then a non-dominated sorting is used to classify the entire population R_t. Though it requires more effort than Q_t alone, but allows a global non-dominated check among the offspring and parent solutions. Once the non-dominated sorting is over, the new population is filled by solutions of different non-dominated fronts, are 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.

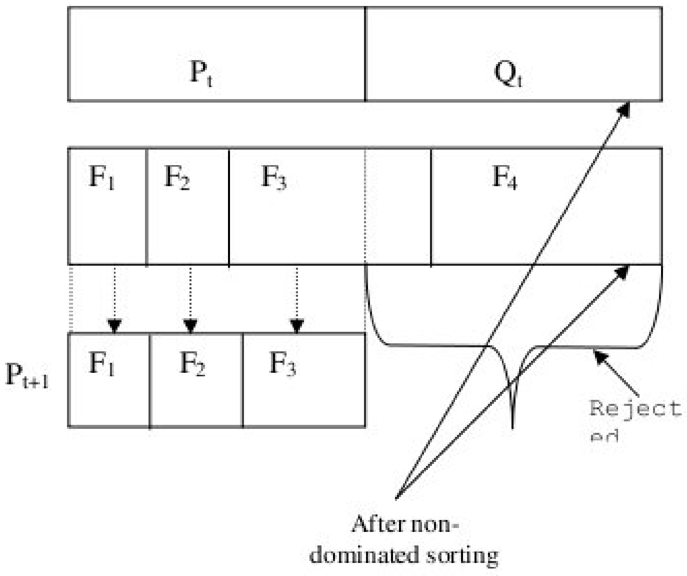

Figure 5. Structural representation of ENSGA

Since the overall population size of R_t is 2N, not all fronts may be accommodating in N slots available in the new population. All fronts, which could not be accommodated, are simply deleted. When the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population. Instead of arbitrarily discarding some members from the last front, it would be wise to use a niching strategy to choose the members of the last front, which reside in the least crowded region in that front.

A strategy like the above does not affect the proceedings of the algorithm much in the early stage of evolution. This is because, early on, there exist many fronts in the combined population. It is likely that solutions of many good non-dominated fronts are already included in the new population, before they add up to n. It then hardly matters which solutions is included to fill up the population. However, during the later stages of the simulation, it is likely that most solution in the population lie in the best non-dominated front. It is also likely that in the combined population R_t of size 2N, the number of solutions in the first non-dominated front exceeds N. The above algorithm then ensures that niching will choose a diverse set of solutions from this set. When the entire population converges to the Pareto-optimal front the continuation of this algorithms will ensure a better spread among the solutions. Following shows the step-by-step format of algorithms.

Algorithm

1. Combine parent and offspring populations and create $R_t = P_t \, U \, Q_t$. Perform a non-dominated sorting to R_t and identify different fronts: F_i, $i= 1,2,\ldots etc$.
2. Set new population $P_{t+1} = 0$. Set a counter $i=1$. Until $|P_{t+1}| + F_i < N$, perform $P_{t+1} = P_{t+1} \cup F_i$ and $i = i+1$.
3. Perform the crowding-sort (F_i, $< c$) procedure (described below) and include the most widely spread ($N- |P_{t+1}|$) solutions using the crowding distance values in the sorted F_i to P_{t+1}.
4. Create offspring Q_{t+1} from P_{t+1} by using the crowded tournament selection, crossover and mutation operators.

Let us see the following crowding-sort (F,<c) procedure:

1. Call the number of solutions in F as $l = |F|$. For each i in the set, first assign $d_i= 0$.
2. For objective function $m = 1, 2, \ldots M$ sort the set in worse order of f_m or, find the sorted indices vector: i^m = sort (f_m , >).
3. For $m= 1,2,.. M,$ assign a large distance to the boundary solutions, or $d_{I1}^{m} = d_{Il}^{m} = \infty$ and for all other solutions j = 2 to l-1, assign
$$d_{jj}^{m} = d_{jj}^{m} + ((f_m(I_{j+1}^{m}) - f_m(I_{j-1}^{m}))/(f_m^{max} - f_m^{min}))$$
The index I_j denotes the solution index of the j th member in the sorted list.

Crowded Tournament Selection Operator

The crowded comparison operator $(<c)$ compares two solutions and returns the winner of the tournament. It assumes that every solution i has two attributes:

1. A non-dominated rank r_i in the population.
2. A local crowding distance (d_i) in the population.

Based on these two attributes, we can define the crowded tournament selection operator as follows: A solution i win a tournament with another solution j if any of the following conditions are true.

1 If solution i has a better rank, i.e. $r_i < r_j$.
2 If they have the same rank but solution i has a better crowding distance than solution j, i.e. $r_i = r_j$ and $d_i > d_j$.

Hence, the one residing in a less crowded area (with a larger crowding distance d_i) wins. The crowding distance d_i can be computed in various ways. In the above we have mentioned a method.

Merits & Demerits

The solutions are competing based on their crowding distances, no niching parameter is required here (such as σ_{share} needed in the MOEA , NSGA's & NPGAs). In the absence of the crowded comparison operator, this algorithm also exhibits a convergence proof to the Pareto-optimal solution set similar to that in Rudolph's algorithm, but the population size would grow with the generation counter. The elitism mechanism does not allow an already found Pareto-optimal solution to be deleted. However when the crowded comparison is used to restrict the population size, the algorithm loses its convergence property.

4.14. Distance Based Pareto Genetic Algorithm (DBPGA)

Osyezka and Kundu (1995) [38][39] suggested an elitist GA a distance–based Pareto genetic algorithm (DPGA), which attempts to emphasize the progress towards the Pareto – optimal front and the diversity along the obtained front by using one fitness measure. This algorithm maintains two populations, One standard GA population P_t where genetic operation are performed and another elite population E_t containing all non-dominated solutions found thus far. The working principle of this algorithm is as follows:

Algorithm

1. Create an initial random population P_0 of size N and set the fitness of the first solution as F_1. Set generation counter t=0.
2. If t=0, insert the first element of P_0 in an elite set E_0 = {1}. For each population member $j \geq 2$ for t=0 and $j \geq 1$ for $t > 0$, perform the following steps to assign a fitness value.
Calculate the distance $dj^{(k)}$ with each elite member k (with fitness $e_m^{(k)}$, m= 1,2…,k) as follows:
$$d_j^{(k)} = \sqrt{\sum ((e_m^{(k)} - f_m^{(j)})/(e_m^{(k)}))^2}$$
Find the minimum distance and the index of the elite member closest to solution j:
$$d_j^{\min} = \min d_j^{(k)},$$
$$K_j^* = \{ k : d_j^{(k)} = d_j^{\min} \}$$
If any elite member dominates solution j the fitness of j is
$$F_j = \max [0, F(e^{(kj^*)} - d^{\min}]$$
otherwise, the fitness of j is
$$F_j = F(e^{(kj^*)}) + d^{\min}$$
and j is included in E_t by eliminating all elite members that are dominated by j.
3. Find the maximum fitness value of all elite members:
$$F_{\max} = \operatorname*{Max}_{k=1}^{|E_t|} F_i$$

All elite solutions are assigned fitness F_{max}.

4. If $t < t_{max}$ or any other termination criteria is satisfied, the process is complete. Otherwise go to step-5.
5. Perform selection, crossover and mutation on P_t and create a new population P_{t+1}. Set $t = t+1$ and go to step-2.

Note that no separate genetic processing is performed on the elite population E_t explicitly. The fitness F_{max} of elite members is used in the assignment of fitness of solutions of P_t.

Merits & Demerits

For some sequence of fitness evaluations both goals of progressing towards the Pareto-optimal front and maintaining diversity among solutions are archived without any explicit niching method.

Since the elite size is not restricted, so it will grow to any size. This will increase the computational complexity of the algorithm with generations. Here the choice of the initial F_i is important. The DPGA fitness assignment scheme is sensitive to the ordering of individual in a population.

5.Future research directions

Although a lot of work has been done in this area, most of it has concentrated on application of conventional or ad-hoc techniques to certain difficult problems. Therefore, there are several research issues that still remain to be solved. Some of which are discussed here.

1. Applicability of MOEA in more difficult real world problems.
2. The stopping criteria of MOGA, because it is not obvious to understand when the population has reached a point from which no further improvement can be reached.
3. Choosing the best solution from Pareto optimal set.
4. Hybridization of multi-objective EAs.
5. Although a lot of work has been done in this area but the theoretical portion is not so much exploited. So a theory of evolutionary multi-objective optimization is much needed, examining different fitness assignment methods in combination with different selection schemes.
6. The influence of mating restrictions might be investigated, although restricted mating is not very widespread in the field of multi-objective EA.

6.Conclusion and Discussion

In this paper, we have described an overview of the EAs such as EP, ES, GP & GA. Then the importance of EA in Multi-objective optimization problems. We attempted to provide a comprehensive review of the most popular evolutionary-based approaches to multi-objective optimization problems, also a brief discussion of their advantages, disadvantages. Finally we discuss the most promising area for future research.

Recently multi-objective Evolutionary algorithms are applied in various fields such as Engineering design problems, Computer networks, Goal Programming, Gas turbine Engine controller design, and resource scheduling etc [40-43].

Other hybridizations typically enjoy the generic and application specific merits of the Individual MOEA techniques that they integrate [44-47].

References

[1] Goldberg, D.E. (1998). "Genetic Algorithms in Search, Optimization, and Machine Learning". Addison-Wesley, Reading, Massachusetts.

[2] Stadler, W. (1984). "A Survey of Multi-criteria Optimization or the Vector Maximum Problem, Part I: 1776-1960". Journal of Optimization Theory and Applications 29, 1(sep), 1-52.

[3] Charnes, A. and Cooper, W.W (1961). "Management Models and Industrial Applications of Linear programming". Vol 1. John Wiley, New York..

[4] Goldberg, D.E., & Deb, K. (1991). "A comparative analysis of selection schemes used in genetic algorithms". In G.J.E. Rawlins (Ed.), Foundations of genetic algorithms (pp. 69-93). San Mateo, CA: Morgan Kaufmann.

[5] Peck, C.C., & Dhawan, A.P. (1995). "Genetic Algorithms as Global Random Search Methods: An alternative perspective". Evolutionary Computation, 3(1), 39-80.

[6] Back, T., Hoffmeister F., and Schwefel H.P., (1992). "A Survey of Evolution Strategies". Proceeding of the Fourth International Conference on Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, 2-9.

[7] Claus, V., Hopf, J. , and Schwefel H.P.(Eds.). "Evolutionary Algorithms and their Application", Dagstuhl seminar Report No- 140, March 1996.

[8] Oyman, A.I. Beyer, H.G. and Schwefel, H.P. (2000). Analysis of the (1,λ)-ES on the Parabolic Ridge", *Evolutionary Computation* 8(3), pages 249-265.

[9] Fogel David B ed.(1998). *Evolutionary Computation* the fossil record, IEEE press.

[10] Koza, J.R. (1992). "Genetic Programming: on the programming of Computers by means of natural Selection". MIT Press.

[11] Koza J.R. (1994). "Genetic Programming II: Autonomous Discovery of Reusable Programs". MIT Press.

[12] Banzhaf W, Nordin P, Keller RE, Francone FD: "Genetic Programming –an Introduction: on the Automatic Evolution of Computer Programs and its Applications". Morgan Kaufmann, 1998.

[13] Evans, G.W. (1984). "An Overview of Techniques for Solving Multi-objective Mathematical Programs". Management Science 30, 11(Nov), 1268-1282.

[14] Horn, J.(1997). "F1.9 Multi-criteria Decision Making". In Back, T., Fogel, D.B. and Michalewicz, Z., editors, Handbook of Evolutionary Computation. Institute of physics Publishing, Bristol, and England.

[15] Ishibuchi, H., Murata, T. (1996). "Multi-objective Genetic Local search algorithm". In T. Fukuda and T. Furuhashi Eds. Proceedings of the 1996 International Conference on Evolutionary computation (Nagoya, Japan, 1996), pp. 119-124. IEEE.

[16] Schaffer, J. D.(1984). "Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Doctoral Dissertation)". Nashville, TN: Vanderbilt University.

[17] Schaffer, J.D.(1984). "Multiobjective Optimization with Vector Evaluated Genetic Algorithms". Unpublished ph.D. Thesis, Vanderbilt University, Nashville, Tannessee.

[18] Schaffer, J.D. (1985). "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms". In Grefenstette, J.J., editor, Proceedings of an International conference on Genetic algorithms and their applications, pages 93-100, sponsored by Texas Instruments and U.S. Navy Center for applied Research in Artificial Intelligence (NCARAI).

[19] Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M.(1998). "Some Guidelines for Genetic Algorithms with Penalty Functions". In J.D. Schaffer Ed., Proceedings of the third International Conference on Genetic algorithms (george Mason University, 1989), pp. 191-197. Morgan Kaufmann Publishers.

[20] Fonseca, C.M. and Fleming P. J. (1993). "Genetic Algorithms for Multi-objective Optimization: Formulation, Discussion, and Generalization". Proceedings of the Fifth International Conference on Genetic Algorithms. 416-423.

[21] Fonseca, C.M. and Fleming, P.J. (1995). "An Overview of Evolutionary Algorithms in Multi-objective Optimization". Evolutionary Computation, 3(1):1-16.

[22] Fonseca, C.M. and Fleming, P.J. (1998). "Multi-objective Optimization and Multiple Constraints Handling with Evolutionary Algorithms-part ii: Application example". IEEE Transactions on Systems, Man, and Cybernetics, 2891):38-47.

[23] Horn, J. and Nafploitis, N. and Goldberg, D.E.(1994). "A Niched Pareto Genetic Algorithm for Multi-objective Optimization". *Proceedings of the first IEEE Conference on Evolutionary Computation*.82-87.

[24] Horn, J. and Nafpliotis, N.(1993). "Multi-objective Optimization using the Niched Pareto Genetic Algorithm". IlliGAL Technical Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbaba, Illinois.

[25] Zitzler,E. and Thiele,L.(1998)."Multi- objective Optimization using Evolutionary Algorithms- A comparative case study".*Parallel problem solving from Nature* ,V, pp 292-301, Springer, Berlin, germany.

[26] Zitzler, E.(199). "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications". Ph. D. Thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. Shaker Verlag, Aachen, Germany, ISBN 3-8265-6831-1.

[27] Zitzler, E. and Thiele, L. (1999). "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach". IEEE Transactios on Evolutionary Computation, 3(4): 257-271.

[28] Deb,k.(1999). "Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems". *Evolutionary Computing Journal*, 7(3), 205-230.

[29] Srinivas, N. and Deb, K.(1994). "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". Evolutionary Computation 2,3 (fall) 221-248.

[30] Srinivas, N. and Deb, K.(1993). "Multi-objective Optimization using Nondominated Sorting in Genetic Algorithms". Technical Report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India.

[31] Kursawe, F.(1991). "A Variant of Evolution Strategies for Vector Optimization". In H.P. Schwefel and R. Manner Eds., parallel problem solving from nature. 1st workshop, PPSN I, volume 496 of Lecturer Notes in Computer Science (Berlin, Germany, oct. 1991), Pp. 193-197. Springer-Verlag.

[32] Hajela, P. and Lin, C.Y.(1992). "Genetic Search Strategies in Multicriterion Optimal Design". Structural optimization 4, 99-107.

[33] Hajela, P. and Lee, J. (1996). "Constrained genetic search via scheme adaptation: An Immune Network Solution". Structural optimization 12, 1, 11-15

[34] Laumanns, M., Rudolph, G., and Schwefel, H. P.(1998). "A Spatial Predator-Prey Approach to Multi-objective Optimization: A Preliminary Study". Proceedings of the Parallel problem solving from Nature, V. 241-249.

[35] Kita, H., Yabumoto, Y., Mori, N., and Nishikawa, Y. (1996). "Multi-objective Optimization by Means of the Thermodynamical Genetic Algorithm". In H.-M. Voigt, W. Ebeling, I Rechenberg, and H.-P. Schwefel Eds., Parallel problem solving from nature-PPSN IV, Lecture Notes in computer science, pp. 504-512. Berlin, Germany: Springer-Verlag.

[36] Rudolph, G.(1994). "Convergence Properties of Canonical Genetic Algorithms". IEEE Transactions on Neural Networks, NN-5, 96-101.

[37] Rudolph, G.(1998). "Evolutionary Search for Minimal Elements in Partially Ordered Finite sets". Evolutionary programming VII. 345-353.

[38] Osyczka, A. and Kundu, S. (1995). "A New Method to Solve Generalized Multi-criteria Optimization Problems using the Simple Genetic Algorithm". Structural Optimization 10, 94-99.

[39] Osyczka, A. and Koski, J. (1982). "Selected Works Related to Multicriterion Optimization Methods for Engineering Design". In Proceedings of Euromech Colloquium (University of Siegen, 1982).

[40] Deb, K. (2002). "Multi-objective Optimization using Evolutionary Algorithms". John Wiley & Sons, Ltd.

[41] Deb, K., Horn, J. (2000). "Introduction to the Special Issue: Multicriterion Optimization". Evolutionary Computation 8(2): iii-iv.

[42] Chipperfield, A.J. and Fleming, P.J. (1995). "Gas Turbine Engine Controller Design using Multi-objective Genetic Algorithms". In A.M.S. Zalzala, editor, *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and applications*, GaLESIA'95,pages 214-219, Halifax Hall, University of Sheffield, UK, IEEE.

[43] Anchor, K.P., Zydallis, J.B., Gunsch, G.H. and Lamont, G.B. (2003). "Different Multi-objective Evolutionary Programming Approaches for Detecting Computer Network Attacks", in Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele (editors), *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pp. 707--721, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal.

[44] Andersson, J. (2003). "Applications of a Multi-objective Genetic Algorithm to Engineering Design Problems", in Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele (editors), *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pp. 737--751, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal.

[45] Jeroen C.J.H. Aerts, Herwijnen, M.V., and Stewart, T.J. (2003). "Using Simulated Annealing and Spatial Goal Programming for Solving a Multi Site Land Use Allocation Problem", in Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele (editors), *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pp. 448--463, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal.

[46] Obayashi, S. (1997). "Pareto Genetic Algorithm for Aerodynamic Design using the Navier-Stokes Equations". In D. Quagliarella, J. Periaux, C.Poloni, and G. Winter, editors, *Genetic Algorithms and Evaluation Strategies in Engineering and Computer Science . Recent Advances and Industrial Applications*, chapter 12, pages 245-266. John Wiley and Sons, West Sussex.

[47] Tan, K.C. and Li. Y. (1997). "Multi-Objective Genetic Algorithm Based Time and Frequency Domain Design Unification of Linear Control Systems". Technical Report CSC-97007, Department of Electronics and Electrical Engineering, University of Glasglow, Glasglow.

[48] Periaux, J., Sefrioui, M., and Mantel, B. (1997). "GA Multiple Objective Optimization Strategies for Eloctromagnetic BackScattering. In D. Quagliarella, J. Periauxx, C. Poloni, and G.Winter, editors, *Genetic Algorithms and Evaluation Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, chapter 11,pages 225-243. John Wiley and Sons, West Sussex.

[49] Belegundu, A.D., Murty, D.V., Salagame, R.R., and Constants E. W. "Multiobjective Optimization of Laminated Ceramic Composites Using Genetic Algorithms". In *Fifth AIAA/USAF/NASA Symposium on Multidisciplinary Analysis and Optimization*, pages 1015-1022, Panama City, Florida, 1994.AIAA.Paper 84-4363-CP.

[50] Abbass, H.A.(2003). "Speeding up Backpropagation Using Multiobjective Evolutionary Algorithms", *Neural Computation*, Vol. 15, No. 11, pp. 2705—2726.

[51] Vedarajan, G., Chan, L.C., and Goldberg, D.E. (1997). "Investment Portfolio Optimization using Genetic Algorithms". In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1997 Conference*, pages 255-263, Stanford University, California, July 1997. Stanford Bookstore.

[52] Poloni, C., and Pediroda, V. (1997). "GA Coupled with Computationally Expensive Simulations: tools to Improve Efficiency". In D. Quagliarella, J.Periaux, C.Poloni, and G. Winter, editors, Genetic Algorithms and Evalution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications, chapter 13,pages 267-288. John Wiley and Sons, West Sussex.

[53] Quagliarella, D., and Vicini, A. (1997). "Coupling Genetic Algorithms and Gradient Based Optimization Techniques". In D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, Chapter 14,pages 289-309.John Wiley and Sons, West Sussex.

About the Author

Ashish Ghosh is an Associate Professor of the Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He received the B.E. degree in Electronics and Telecommunication from the Jadavpur University, Calcutta in 1987, and the M.Tech. and Ph.D. degrees in Computer Science from the Indian Statistical Institute, Calcutta in 1989 and 1993, respectively. He received the prestigious and most coveted *Young Scientists* award in Engineering Sciences from the Indian National Science Academy in 1995; and in Computer Science from the Indian Science Congress Association in 1992. He has been selected as an *Associate* of the Indian Academy of Sciences, Bangalore in 1997. He visited the Osaka Prefecture University, Japan with a Post-doctoral fellowship during October 1995 to March 1997; and Hannan University, Japan as a *Visiting scholar* during September-October, 1997. During May 1999 he was at the Institute of Automation, Chinese Academy of Sciences, Beijing with CIMPA (France) fellowship. He was at the German National Research Center for Information Technology, Germany, with a German Govt. (DFG) Fellowship during January-April 2000. During October-December 2003 he was a *Visiting Professor* at the University of California, Los Angeles. He also visited various Universities/Academic Institutes and delivered lectures in different countries including South Korea, Poland, and The Netherland. His research interests include *Evolutionary Computation, Neural Networks, Image Processing, Fuzzy Sets and Systems, Pattern Recognition, and Data Mining*. He has already published about 55 research papers in internationally reputed journals and referred conferences, has edited 4 books, and is acting as guest editor of various journals.

Satchidananda Dehuri is currently working as a lecturer in Silicon Institute of Technology, India. He received his M.Sc. (Mathematics) from Sambalpur University in 1998, and M.Tech. (Computer Science) from Utkal University in 2001. He is working toward the Ph.D. degree in the Department of Computer Science and Application of Utkal University. He has published 10 papers in various International Conferences. His research interests include pattern recognition, evolutionary algorithm for multi-criterion optimization, and Data Mining & Data Warehousing