2

Linear Evolutionary Algorithm

Kezong Tang[1,2], Xiaojing Yuan[2], Puchen Liu[3] and Jingyu Yang[1]
[1]*Computer Science and Technology Department Nanjing University of Science and Technology,*
[2]*Engineering Technology Department University of Houston,*
[3]*Department of Mathematics University of Houston,*
[1]*China*
[2,3]*United States*

1. Introduction

During the past three decades, global optimization problems (including single-objective optimization problems (SOP) and multi-objective optimization problems (MOP)) have been intensively studied not only in Computer Science, but also in Engineering. There are many solutions in literature, such as gradient projection method [1-3], Lagrangian and augmented Lagrangian penalty methods [4-6], and aggregate constraint method [7-9]. Among these methods, penalty function method is an important approach to solve global optimization problems.. To obtain the optimal solution of the original problem, the first step is to convert the optimization problem into an unconstrained optimization problem with a certain penalty function (such as Lagrangian multiplier). As the penalty multiplier approaches zero or infinite, the iteration point might approach optimal too. However, at the same time, the objective function of the unconstrained optimization problem might gradually become worse. This leads to increased computational complexity and long computational time in implementing the penalty function method to solve the complex optimization problems. In most of the research, both the original constraints and objective function are required to be smooth (or differentiable). However, in real-world problem, it is seldom to be able to guarantee a derivative for of the specific complex optimization problem. Hence, the development of efficient algorithms for handling complex optimization problems is of great importance. In this chapter, we present a new framework and algorithm that can solve problems belong to the family of stochastic search algorithms, often referred to as evolutionary algorithms.

Evolutionary algorithms (EAs) are stochastic optimization techniques based on natural evolution and survival of the fittest strategy found in biological organisms. Evolutionary algorithms have been successfully applied to solve complex optimization problems in business [10,11], engineering [12,13], and science [14,15]. Some commonly used EAs are Genetic algorithms (GAs)[16], Evolutionary Programming (EP)[17], Evolutionary Strategy (ES)[18] and Differential Evolution (DE)[19]. Each of these methods has its own characteristics, strengths and weaknesses. In general, a EA algorithm generate a set of initial solutions randomly based on the given seed and population size. Afterwards, it will go through evolution operations such as cross-over and mutation before evaluated by the

objective function. The winning entity in the population will be selected as the parents (or seed) of the next generation (i.e., iteration). The optimization iteration continues until the termination criteria are satisfied. Typically, either the evolution process reached user defined maximum number of iteration or the improvement in objective function between the two generations converges.

The major advantages of the improved EAs compared with traditional optimization techniques include [20-23]:

1. EAs do not require objective function to be continuous and can be used in algebraic form.
2. EAs tend to escape more easily from local optimum due to the randomness introduced at the beginning and perturbation introduced by the mutation operation. The amount of perturbation is a parameter defends on the step size specified by the user.
3. EAs do not require specific domain information or prior knowledge although they can exploit it if such information is available. It does not involve calculation of the gradients of the objective function.
4. EAs are conceptually simple and relatively easy to implement.

The major disadvantages of EAs are their poor performance in handling constraints, long computational time, and high computational complexity, especially when the solution space is hard to explore. To overcome these difficulties, some 'more intelligent' rules and /or hybrid techniques such as evolutionary-gradient search (EGS) have been developed to extend EAs to overcome the slow convergence phenomena of the EAs near the optimum solution [24-27]. In addition, improving fitness function, crossover and mutation operators, selection mechanisms, and adaptive controlling of parameter settings all enhance EA's efficiency and performance. An excellent comparison study of evolutionary algorithms has been published for global optimization problems by Michalewicz and Schoenauer [28].

Among the evolutionary algorithms the methods based on penalty functions have proven to be the most popular. These methods augment the cost function, so that it includes the squared or absolute values of the constraint violations multiplied by penalty coefficients. However, there are also serious drawbacks with penalty function methods. For example, small values of the penalty coefficients drive the search outside the feasible region and often produce infeasible solutions [29], if imposing very severe penalties makes it difficult to drive the population to the optimum [29-31]. To overcome these drawbacks, Kim and Myung [26] proposed the concept of two phase evolutionary algorithm, where the penalty method is implemented in the first phase, while during the second phase an augmented Lagrangian function is applied on the best solution of the first phase. Tahk and Sun [32] presented the co-evolutionary augmented Lagrangian method which uses an evolution of two populations with opposite objectives to solve constrained optimization problems. Tang proposed a special hybrid genetic algorithm (HGA) [33]with penalty function and gradient direction search, which uses mutation along the weighted gradient direction as the main operation and only in the later generation it utilizes an arithmetic combinatorial crossover. The approach presented in [34] is an extended hybrid genetic algorithm (EHGA), which is a fuzzy-based methodology that embeds the information of the infeasible points into the evaluation function.

Based on the above analysis, our major concern in this chapter was how to design a linear fitness function based on the general penalty function so as to fast evaluate candidate solutions, regardless of the design variables' dimensions of solving the complex optimization problems. The major advantage of linear function is their simplicity and

computational attractiveness. The chapter starts with the review that we have a briefly review for various evolutionary approaches in the last years. In the sequel we will focus on this compression process, in which search space of design variables will be compressed into 2-dimensional performance space and it is possible to fast discriminate 'good' solutions from candidate solutions, regardless of the complexity of original space. In addition, our method combines two improved operators in reproduction phase, i.e., crossover and mutation. Simulation results over a comprehensive set of benchmark functions show that our method is feasible and effective. Meanwhile, it can provide good performance in terms of uniformity and diversity of solutions.

2. Description of EA

Evolutionary algorithm is a random search based optimization technique. Various applications have shown that when problems are formulated properly, EA can give good results with reasonable time complexity. EA mimics the process of natural selection and starts with artificial individuals (represented by a population of "chromosomes"). EA tries to evolve those individuals that are fitter and, by applying genetic operators (crossover and mutation), it attempts to produce descendants that are better than their parents in terms of a certain quantitative measure. In spite of their diversity, most of them are based on the same iterative procedure.

As a heuristic population-based method, EA is really like a "black box", completely independent from the characteristic of the problem. Fig.1 presents the classical EA flow chart. An initial population of individuals is generated randomly. Each of these individuals is evaluated in terms of a certain "fitness function" that can "guide" EA to the desired region of the search space. EA's three genetic operators (Selection, Crossover and Mutation) are the main components to improve the EA's behavior. Selection is the process that mimics the "survival of the fittest" principle in the biological theory of evolution. Firstly, the selection operator assures that individuals are copied to the next generation with a probability associated to their fitness values. Although selection is implemented in a EA as a policy for determining the best candidate individuals that will be presented in the next generation with a higher probability, it does not search the space further, because it just copies the previous candidate individuals. The search results from the creation of new individuals from old ones. Secondly, the crossover operator is implemented in EA by exchanging chromosome segments between two randomly selected chromosomes.

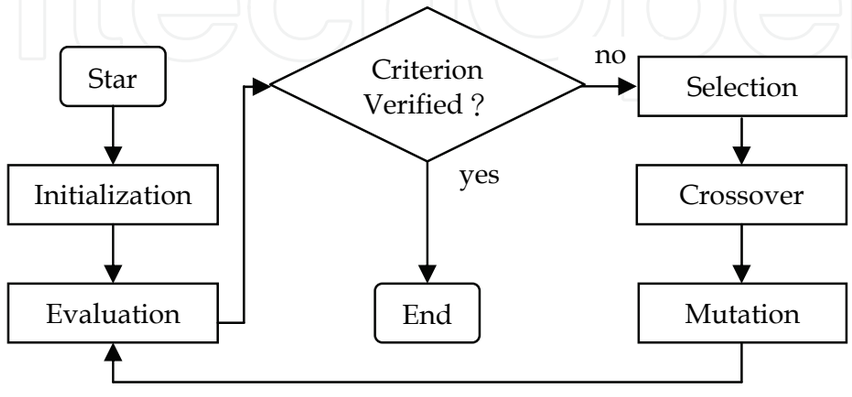


Fig. 1. Evolutioary algorithm flow chart

Crossover process provides a mechanism to allow new chromosomes to inherit the properties from old ones. Thirdly, mutation is a random perturbation to one or more genes in the chromosomes during evolutionary process. The purpose of the mutation operator is to provide a mechanism to avoid local optima by exploring the new regions of the search space, which selection and crossover could not fully guarantee. The searching process terminates when the predefined criterion is satisfied.

3. Related concepts of MOP

Almost all real world engineering designing problems are characterized by the presence of several conflicting and/or cooperating objectives, as opposed to having a single objective, and result in a set of non-dominated solutions. This set, generally called Pareto front, helps the decision-maker to identify the best compromise solutions by eliminating inferior ones and articulating his preference pertaining to the different objectives once he has an additional knowledge of the Pareto frontier. The term MOP is used to broadly classify problems with more than one objective. Without loss of generality, a general multi-objective optimization problem can be expressed in the following equations:

$$\min F(x) = (f_1(x), f_2(x), ..., f_k(x))^T \tag{1}$$

$$s.t. G(x) = (g_1(x), g_2(x), ..., g_l(x))^T \le 0 \tag{2}$$

$$H(x) = (h_{l+1}(x), h_{l+2}(x), ..., h_m(x)) = 0 \tag{3}$$

where k is the number of objective functions, l and m-l are the number of unequal and equal constraints respectively, and the vector G(x) represents constraints that probably are easily handled explicitly, such as lower and upper bounds on the variables, $x=(x_1,x_2,\ldots,xn) \in S \subseteq \Omega$. n is the number of designing variables. Ω and S are the searching space of objective function and the feasible searching space, respectively.

The Pareto optimal concept is initially introduced by Vilfredo Pareto in the 19th century, and the concept has already been widely used in MOP to aid designers in their decision-making processes. In this paper, we assume that all objectives are to be minimized for clarity purpose since maximization of any maximization of any $-f(\bullet)$. The Pareto optimal concept is stated as follows[35,36]:

Definition 1 order relation between design vectors. Let x and x' be two designing variables. The dominance relations in a minimization problem are:

x dominates x' ($x \prec x'$), *iff* $f_t(x) < f_t(x')$ and $f_{t'}(x) \not> ft'(x')$, $\forall\ t' \neq t \in [1, k]$.

x are incomparable with x' ($x \sim x'$),*iff* $f_t(x) < f_t(x')$ and $f_{t'}(x) > ft'(x')$, $t' \neq t \in [1, k]$.

Definition 2 Pareto-optimal solution. A solution x is called Pareto-optimal if there is no other $x' \in F$, such that $f(x') < f(x)$. All the Pareto-optimal solutions define the Pareto-optimal set.

Definition 3 Non-dominated solution. A solution $x \in S$ is non-dominated with respect to a set $x' \in S$ if and only if $\nexists\ x' \in S$, verifying that $x' \prec x$.

Definition 4 Non-dominated set. Given a set of solutions S', such that $S' \in S$ and $Y'=f(S')$, the function $h(S')$ returns the set of non-dominated solutions from S':

$$h(S')=\{\ \forall\ x \in S'\ |\ x \text{ is non-dominated by any other } z',\ z' \in S'\ \}$$

Fig.2 graphically describes the process of mapping from designing space to objective space with objectives (f_1 and f_2), and the Pareto-optimal set are shown as the Pareto optimal front. All of the Pareto solutions in designing space are equally important and all are the global optimal solutions. The decision-maker articulates his preference pertaining to the different objectives once he has knowledge of the Pareto front.

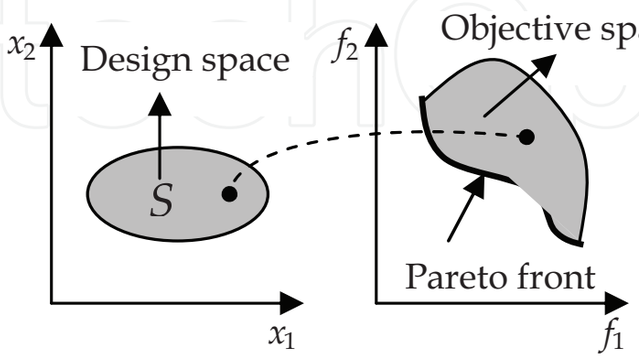


Fig. 2. Mapping from design space to objective space.

4. The linear fitness function (LFF)

A very popular approach for handling complex constraints by using EAs is to adopt penalty function such as [37-39]. When handling individuals violating any one of the constraints, if the added penalties do not depend on the current iteration number and remain constant during the entire evolutionary process, then the penalty function is called static penalty function, and its penalties are weighted sum of all constraint violations. If, alternatively, the current iteration number is considered while determining the penalties, then the penalty function is called dynamic penalty function. In this paper, we adopt static penalty function as the following manner:

$$p_j(x)=\begin{cases}\max(0, g_j(x)), & 1 \le j \le q \\ |h_j(x)|, & l+1 \le j \le m\end{cases} \tag{4}$$

where $p_j(x)$ denotes the degree of individual violating constraints. The generally fitness function for evaluating individuals can be defined as below:

$$fitness(x) = f(x) + r \times \sum_{j=1}^{m} p_j(x) \tag{5}$$

$$f(x) = \sum_{i=1}^{k} w_i f_i(x) = w^T F(x),$$

where $f(x)$ is a convex combination of the different objectives in that the multi-objective problem is converted into a scalar optimization one. The weighted value w_i is chosen such that $w_i \ge 0$, i=1,…,n, and $\sum_{i=1}^{k} w_i = 1$.

One of existing difficulties in penalty function is mainly that parameter r is not easily to be selected and controlled [37]. For many MOP, we note that the searching space of the design

vector is always situated in *n*-dimension space ($n \geq 2$). In terms of human imagination of space, if we can attempt to give a good method to transform *n*-dimension space into low dimensional space for MOP since the dimensions below three are geometrically prone to human understanding. Therefore, we give the following transforming procedure.

$y_1 = f(x), y_2 = \sum_{i=1}^{m} p_i(x)$. If we can make an appropriate mapping between vector $y = [y_1, y_2]$ and $\bar{x}$, then *fitness*(x) can be represented as a linear fitness function:

$$fitness(\bar{x}) = a^T \bar{x} = \sum_{i=1}^{2} a_i \bar{x}_i, \tag{6}$$

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ r \end{bmatrix}, \bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y$$

without loss of generality, the variable $\bar{x}$ is still denoted by x. x is a point in 2-dimension space. $a^T y = 0$ ascertains a hyperplane via origin, then the entire search space is divided into two subspaces with 2-dimension respectively. a is a normal vector in hyperplane (see Fig.3). The division of initial searching space with *n*-dimension is equivalent to the above results, and the searching process is focused on Φ_1. A mapping process is described between feasible region *S* and corresponding abnormity region Φ_3 while any point x in 2-dimension space is required to satisfy one of the following expressions:

$$fitness(x) \begin{cases} > 0, x \in \Phi_1, \\ < 0, x \in \Phi_2, \\ = 0, x \in H, \end{cases} \tag{7}$$

Further analysis shows that linear fitness function may be regarded as an algebraic measurement from point x to the hyperplane.

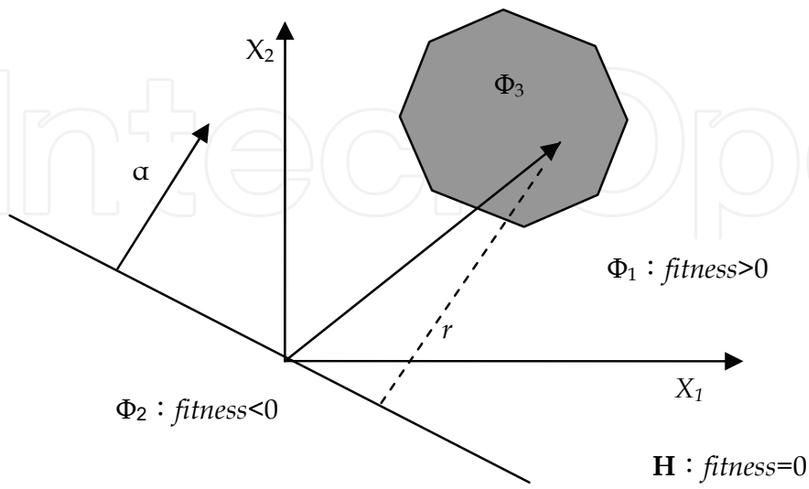


Fig. 3. Transformation of searching space.

By means of above anaysis, we give a new linear fitness function to evaluate individuals:

$$fitness(x) = \frac{a^T x}{||a||} \tag{8}$$

While evaluating new individuals we may directly use formula (8) without considering the feasibility of individual. It is very convenient to identify the good or bad individuals from population according to corresponding fitness values.

5. Evolutionary operators

In conventional EAs, the basic operators are selection and mutation. The selection assigns greater probabilities to the fittest individuals according to its fitness. Crossover is the exchange of information between different individuals, and it is the principal process in generating new individuals. Mutation is a security factor to avoid entrapment at any other point when the population is completely converged. The parents undergo crossover and mutation to generate two new children, and each individual in the population evolve to get higher generation by generation.

5.1 Crossover operator

The crossover operator based on a modified self-adaptive density adopts two parents to generate children [40]. The procedure is formulated as follows:

First, two individuals are selected using a random selection procedure to generate new individuals. The two parents can be expressed as $X=(x_1, x_2, \ldots, x_d)$ and $Y=(y_1,y_2,\ldots,y_d)$, where $x_i < y_i$.

$$[x_i + \frac{x_i - y_i}{2}, y_i + \frac{y_i - x_i}{2}] = [\bar{x}_i, \bar{y}_i] \tag{9}$$

Here, we give a density function in (10). It depends on two parameters a and β, and a, $\beta \in [0,1]$.We give a cumulated distribution function in (11) and $0 \le G_{x_i,y_i} \le 1$. It ascertains parameter η_i in children by (12). $G^{-1}_{x_i,y_i}$ is the inverse function of G_{x_i,y_i} in (12).

$$g_{x_i,y_i}(x,\alpha,\beta) : [\bar{x}_i, \bar{y}_i] \times \alpha \times \beta \to r \tag{10}$$

$$G_{x_i,y_i}(x,\alpha,\beta) : [\bar{x}_i, \bar{y}_i] \times \alpha \times \beta \to [0,1] \tag{11}$$

$$\eta_i = G^{-1}{}_{x_i,y_i}(r,\alpha,\beta) \tag{12}$$

The experimental results show that density crossover operator is good at finding optimal solutions and enlarging the search region.

5.2 Mutation operator

The neighborhood mutation operator is introduced in Ref.[41]. Individual mutation is in its neighborhood space by using $x'=x+a\times r$, where x' and x are child and parent respectively, a is a uniformly random number in [-1,1], and r is the radius of neighborhood space dynamically compressed using $r=r\times cr$, where $cr=(1+0.1^b\times c)^{-t}$ and cr is the compression ratio, b and c are random integer between 0 and 9, t is the current generation.

Although the neighborhood mutation operator evenly scans the whole neighborhood space of individual at the beginning, the exploration becomes localized with the generation increasing. In order to overcome this deficiency, the following improvements are made on the neighborhood mutation operator. Give a binary number δ, add the following part.

$$x'_{new} = \begin{cases} x' + \lambda(t, b_i - x'), if \quad \delta = 0 \\ x' - \lambda(t, x_i - a_i), if \quad \delta = 1 \end{cases} \tag{13}$$

$$\lambda(t, \tau) = \tau(1 - r^{\gamma}), \gamma = (1 - t / n_{\max})^{\beta}$$

where r is a random number in [0,1], $n_{\max}$ and β denote total iterative number and random number respectively. The added part uniformly explores the whole searching space during the execution of the algorithm. Hence, it may overcome the deficiency of the neighborhood mutation operator. By integrating the above two parts, not only the abilities of the global search and local exploration are balanced, but also the diversity of the population increases. As a result, the premature convergence is avoided in a way. The experimental results show that the improved mutation operator is very feasible for the known searching space and insure x'_{new} is a random number in $[a_i, b_i]$.

6. Numerical experiments

6.1 Linear evolutionary algorithm

The LEA has a different design from other variants of EAs. It does not use any information from the dominated individuals. In each generation, we only preserve nondominated individuals. The number of generated children is a fixed constant η. However, we limit the number of the nondominated individuals using a nearest neighborhood distance function (NNDF) in Ref.[41], which can help disperse the non-dominated individuals. The LEA is described as follows:

Step 1. (***Initialization***). Generate an initial population containing N_{pop} individuals where N_{pop} is the number of individuals in each population.

Step 2. (***Evaluation***). Calculate the fitness values of the generated individuals using the LFF. Update a tentative set of non-dominated solutions. The number of non-dominated solutions is $u_{nd.}$

Step 3. (***Selection***). If $u_{nd} > u_{max}$, select u_{max} individuals using NNDF, then $u_{nd} = u_{max}$.

Step 4. (***Crossover and mutation***). Generate the η offspring using crossover based on density and modified neighbourhood space mutation.

Step 5. (***Termination test***). If a prespecified stopping condition is not satisfied, return to step 2.

In step 3, we control the number of non-dominated individuals so as not to exceed a maximum number, u_{max}. Hence the generated offspring are under control. To filter better individuals from the tentative set, we evenly distribute the individuals on the Pareto front using NNDF.

Consider the entire complexity of one iteration of the proposed algorithm, the overall complexities of the algorithm are focused on evaluation and selection. Assuming the solved optimization problems totally have m decision variables, and population size is n. The basic operations and their worst-case complexities are as follows:

1. Evaluation in step 2 $o(2n(n+m))$.
2. Selection in step 3 $o(n^2)$.

The overall complexity of the algorithm is $o(2n(n+m))$, which is governed by evaluation in step2 (including linear mapping for decision variables from n-dimension to 2-dimension and updating a tentative set of non-dominated solutions).

6.2 Simulation results

Simulations are performed in MATLAB with a 2 GHZ Pentium PC. In our study, five numerical constrained optimization problems were divided into two groups ($G1$ and $G2$) were applied to test the LEA. These benchmark problems are taken from Ref.[37] and [41]. $G1$ only contains inequality constraints in test problems which involving two designing variables. $G2$ contains inequality and equality constraints of single objective optimization problems which have no limitation on the number of designing variables. For all conducted experiments, four parameters of the LEA, namely, population size (Pop_t), iterative number (NG), Crossover probability (CP) and mutation probability (MP) are set 200, 300, 0.9, 0.05. All problems are repeated for 200 in the same environment.

$G1$: *Test Problem 1 BNH*

$$\min F_1(x) = (f_1(x), f_2(x)); f_1(x) = 4x_1^2 + 4x_2^2; f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2$$

$$s.t.\ C_1(x) = (x_1 - 5)^2 + x_2^2 \le 25; C_2(x) = (x_1 - 8)^2 + (x^2 + 3)^2 \ge 7$$

$$0 \le x_1 \le 5, 0 \le x_2 \le 3\ .$$

Test problem 2 TNK

$$\min F_2(x) = (f_1(x), f_2(x)); f_1(x) = x_1; f_2(x) = x_2;$$

$$s.t.\ C_1(x) = x_1^2 + x_2^2 - 0.1\cos(16\arctan(x_1 / x_2)) - 1 \ge 0;$$

$$C_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \le 0.5\ ;$$

$$0 \le x_1 \le \pi, 0 \le x_2 \le \pi\ .$$

Test problem 3 Constr-Er

$$\min F_3(x) = (f_1(x), f_2(x)); f_1(x) = x_1; f_2(x) = (1 + x_2) / x_1;$$

$$s.t.\ C_1(x) = 9x_1 + x_2 - 6 \ge 0; C_2(x) = 9x_1 - x_2 - 1 \ge 0;$$

$$0.1 \le x_1 \le 1, 0 \le x_2 \le 5\ .$$

Here, the circular symbols represent the results obtained using NSGA-II or LEA in six figures. For case 1, *BNH* is a two-objective function problem with a convex Pareto front and constrained conditions are two inequalities. LEA gives a very good approximation of the Pareto front by obtaining evenly distributed solutions, as shown in Fig.5. However, Fig.4

shows that although a number of optimal solutions are obtained using NSGA-II, in terms of diversity of solutions, these solutions are not evenly spread out over the entire front. There exists some disconnected spaces on Pareto front in Fig.4. The Pareto front consists of $x_1^* = x_2^* \in [0,3]$, $x_1^* \in [3,5]$ and $x_2^* = 3$. For case 2, constraint conditions are also two inequalities' state. The Pareto front is well predicted, and a number of optimal solutions obtained are spread out over the entire front using the LEA in Fig.7. Decision makers make a final choice of optimal solution according to real conditions from Pareto optimal set. But result of Fig.6 shows that we probably are not able to well predict the three disconnected curves. For case 3, it is the two-objective function problem of Constr-Ex with two-dimensional curve. Fig.9 shows the predicted Pareto front in the two-dimensional objective space obtained by the LEA. The method can capture the distinct solution along this front. Fig.8 shows that comparative method can performs well elsewhere along the Pareto front.

We adopt some performance measures described in [42] so as to obtain more quantitative measures of algorithm performances. These performance metrics are generational distance and the diversity metric. The performance of algorithm is measured both in terms of the proximity to the true Pareto front that is achieved as well as in terms of the diversity of the optimal solutions. The means and variance of these measures are evaluated by conducting 20 distinct runs of each simulation. The results are tabulated in table 1. From table 1, we can see that the LEA has better convergence performance than NSGA-II because it is obvious that the generational distance metric is larger than the later, and diversity metric is also larger using NSGA-II. The LEA is an effective and robust method.

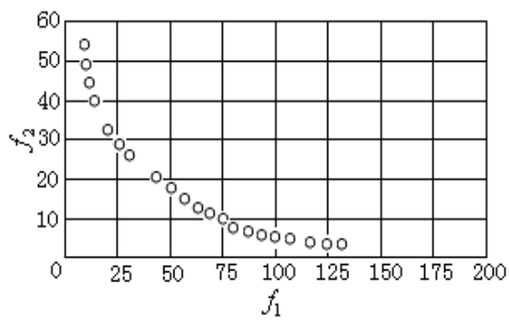


Fig. 4. Pareto front on BNH using NSGA-II

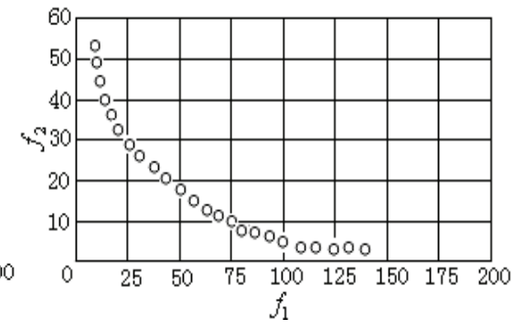


Fig. 5. Pareto front on BNH using LEA

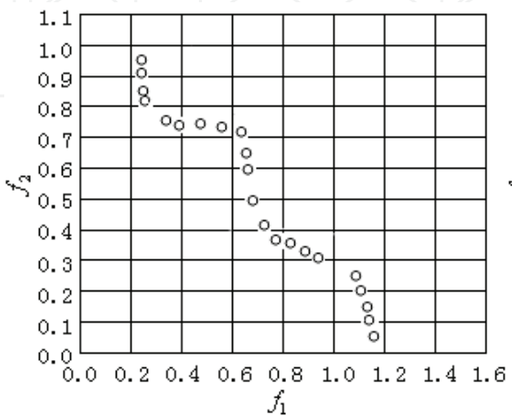


Fig. 6. Pareto front on TNK using NSGA-II

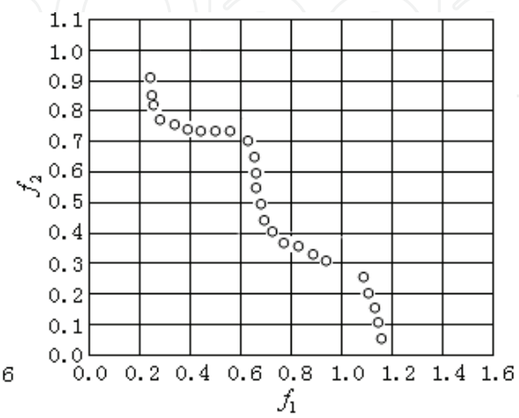


Fig. 7. Pareto front on TNK using LEA

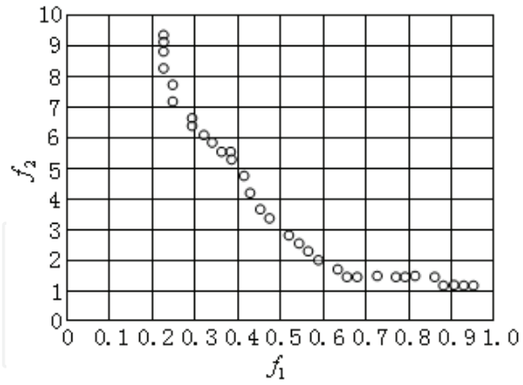


Fig. 8. Pareto front on CE using NSGA-II

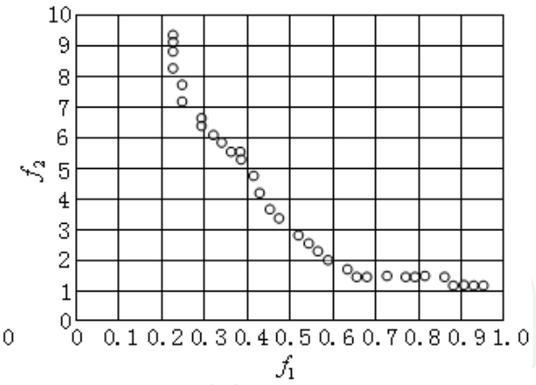


Fig. 9. Pareto front on CE using LEA

Test problems	Generational distance		Diversity	
	Mean	Variance	Mean	Variance
BNH	0.0032356	0.0000009	0.4516783	0.00156753
	0.0026733	0.0000005	0.4356745	0.00107867
TNK	0.0051289	0.0000012	0.2389783	0.00825676
	0.0050125	0.0000011	0.2337892	0.77867826
CE	0.0043216	0.0000011	0.3567882	0.00578933
	0.0040102	0.0000010	0.3389563	0.00623124

Table 1. The results of mean and variance on test problems using NSGA-II and LEA respectively.

G2: Test Problem 2 TP1

$$\min g(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

$$s.t.\ \ x_4 - x_3 + 0.55 \geq 0, -x_4 + x_3 + 0.55 \geq 0 \ ;$$

$$1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0;$$
$$1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0;$$
$$1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

$$0 \leq x_i \leq 1200(i = 1,2) \ ; \ 055 \leq x_i \leq 0.55(i = 3,4) \ .$$

Test Problem 2 TP2

$$\min g(x) = e^{x_1 x_2 x_3 x_4 x_5}$$

$$s.t.\ x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \ ;$$

$$x_2 x_3 - 5x_4 x_5 = 0, x_1^3 + x_2^3 + 1 = 0 \ ;$$

$$-2.3 \leq x_i \leq 2.3(i = 1,2) \ ; \ -3.2 \leq x_i \leq 3.2(i = 3,4,5) \ .$$

Two test problems in second group contain equality and inequality's constraints that are tested for single objective optimization problems. In many real-world optimization problems, optimal solutions of them are obtained by transforming MOP into single objective optimization. Thus, the research of single objective optimization is also very improtant in engineering application areas. For *TP*1, we find total optimal solutions for 25, (668.94675327911, 1013.10377656821, 0.10773654866, 0.39654576851). Distance of optimal solution is 2.3323246783310e-13, which is corresponding to optimal value 5198.5467. For test problems 2, we find total optimal solutions for 21, x = (-1.77365745561,1.45675698761,-1.5678457772,0.66755656893,-0.75778765788), which is corresponding to the optimal value 0.055894567. The mean and worst values of solutions are formulated for two test problems in Table 2.

For *TP*1 and *TP*2, results of the first row are obtained using LEA, similarly, the second row takes a Pareto strength evolutionary algorithm [37] (denoted by ZW). The third row means results of a random sorting [43] (denoted by RY). From Table 2, we can see that the LEA outperforms other two algorithms in terms of experimental data involving the best value, mean and the worst value, as demonstrate the LEA is a robust algorithm with generality and effectivity.

Problems		Best	Mean	Worst
*TP*1	LEA	5126.4266	5126.5461	5126.9586
	ZW	5126.49811	5126.52654	5127.15641
	RY	5126.497	5128.881	5142.472
*TP*2	LEA	0.053945563	0.053999775	0.054993677
	ZW	0.053949831	0.053950257	0.053972292
	RY	0.053957	0.057006	0.216915

Table 2. Comparison among LEA(new algorithm), ZW(in Ref.[27]) and RY(in Ref.[33]) (40 independent run)

7. Conclusion

In this paper, we propose an approach of using the LEA to optimize MOP, which finds the optimal solutions using the method of transforming the search space with high dimensions into low dimensional space. Numerical experiments show that the LEA performed well in the problems of two groups in terms of the quality of the solutions found. Moreover, the LEA is also a fast and robust method. A future work aims to validate this new optimization method on real-life engineering optimiztion problems (e.g. issued from mechanical engineering and power systems).

8. References

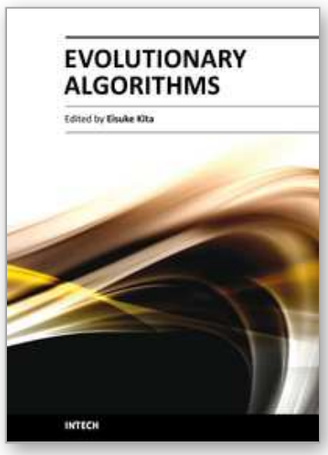
[1] Yang, J.B. (1999). Gradient projection and local region search for multiobjective optimisation. *European Journal of operational Research*. 112(2), 432-459.

[2] Harada, K. ; Sakuma, J.; Ono, I.; Kobayashi, S. (2007). Constraint-handling method for multi-objective function optimization: Pareto descent repair operator. *In 4th International Conference of Evolutionary Multi-Criterion Optimization*. pp.156-170.

[3] Miettinen, K.; Makela, M. M.(1993). Interactive method for nonsmooth multiobjective optimization with an application to optimal control. *Optimization Methods and Software*. 2(1):31-44.

[4] Bazaraa,M.S.; Shetty,L.M.(1993). *Non-linear programming: theory and algorithms*. New York:Wiley.
[5] McCormick, G.P. (1983). *Nonlinear programming: theory,algorithms and applications*.New York: Wiley.
[6] Zhang, J.Z.; Zhang, L.W.(2010). An augmented Lagrangian method for a class of inverse quadratic programming problems. *Applied Mathematics and Optimization*. 61(1):57-83.
[7] Yu, B.; Feng, G.C.; Zhang, S.L.(2001).The aggregate constraint homotopy method for nonconvex nonlinear programming. *Nonlinear Analysis*. 45(7):839-847.
[8] Thakur, M.; Wang, L.Z.; Hurhurgh, C.R. (2010). A multi-objective optimization approach to balancing cost and traceability in bulk grain handling. *Journal of Food Engineering*. 101(2):193-200.
[9] Li, Y.Y.; Tan, T.; Li, X.S.(2010). A gradient-based approach for discrete optimum design. *Structural and Multidisciplinary Optimization*. 41(6):881-892.
[10] Su, C.T.; Chiang, C.L.(2004). An incorporated algorithm for combined heat and power economic dispatch. *Electric Power Systems Research*. 69(2-3):187-195.
[11] Wu, J. Y. (2010). Computational Intelligence-Based Intelligent Business Intelligence System: Concept and Framework.*In Proceedings of the 2010 Second International Conference on Computer and Network Technology*. pp. 57-89.
[12] Carlos,A.; Coello,C.(2000). Use of a self-adaptive penalty approach for engineering optimization problems. *computers in Industry*.41(2):113-127.
[13] Falconer, M.; Greenwood, G.; Morgan, K.;et al. (2010). Using evolutionary algorithms for signal integrity assessment of high-speed data buses. *Journal of Electronic Testing: Theory and Applications*. 26(3): 297-305.
[14] Rahnamayan, S.; Tizhoosh, H.R.; Salama,M.M.A. (2007). A novel population initialization method for accelerating evolutionary algorithms.*Computers and Mathematics with Application*. 53(10):1605-1614.
[15] Nobakhti, A.(2010). On natural based optimization. *Cognitive Computation*. 2(2):97-119.
[16] Ge, J.K.; Qiu, Y.H.; Wu, C.M.; et al. (2008). Summary of genetic algorithms research. *Application Research of Computers*. 25(10):2911-2916. (In Chinese).
[17] Dong, H.B.; He, J.; Huang, H.K.; et al. (2007). Evolutionary programming using a mixed mutation strategy. *Information Sciences*. 177(1):312-327.
[18] Jürgen, P.(2004). Finding optimal decision scores by evolutionary strategies. *Artificial Intelligence in Medicine*. 32(2):85-95.
[19] Babu, B.V.; Chakole, P.G.; Syed Mubeen, J.H.(2005). Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor. *Chemical Engineering Science*. 60(17): 4822-4837.
[20] John w. Fowler,; Esma S. Gel.; Murat M.K□ksalan. ;et al.(2010). Interactive evolutionary multi-objective optimization for quasi-concave preference functions. *European Journal of Operational Research*. 206(2):417-425.
[21] Sun, D.Y.; NI, S.C.; Huang, T.C.(2010). Apply a novel evolutionary algorithm to the solution of parameter selection problems. *Applied Mathematics and Ccomputation*. 216(11):3343-3354.
[22] Mitchell, M. *An introduction to genetic algorithms*. Cambridge, MA:MIT Press,1996.
[23] Sarimveis, H.; Nikolakopoulos, A. (2005). A line up evolutionary algorithm for solving nonlinear constrained optimization problems. *Computers & Operations Resarch*. 32(6): 1499-1514.
[24] Glannakoglou,K.C.; Papadimitriou, D.I.; Kampolis, I.C.(2006).Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Compute Methods in Applied Mechanics and Engineering*. 195(44-47):6312-6329.

[25] Rada. R.(1982). Evolutionary search: Gradients and information.*Biosystems*. 15(2):169-177.
[26] Kim J. H, Myung H. ; Evolutionary programming techniques for constrained optimization problems. IEEE Trans Evol Comput 1997;1(2):129-140.
[27] Salomon R. Evolutionary algorithms and gradient search: similarities and differences. IEEE Transa Evol comput 1997;2(2):45-55;
[28] Michalewicz, Z.; Schoenauer, M.(1996). Evolutionary algorithms for constraint parameter optimization problems. *Evolutionary Computation*. 4(1):1-32;
[29] Michalewicz, Z. *Genetic algorithms + data structures = evolution programming*. New York: Springer,1996.
[30] Anderson,E. J.; Ferris, M. C.(1994). Genetic algorithms for combinatorial optimization: the assembly line balancing problem. *ORSA Journal on Computing*.6(2):161-73..
[31] Coit, D. W.; Smith, A.E.; Tate, D.M. (1996). Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*. 8(2):173-82.
[32] Tahk, M.J.; Sun, B.C. (2000). Co-evolutionary augmented Lagrangian methods for constrained optimization. IEEE Transactions on Evolutionary Computation . 4(2):114-24.
[33] Tang, J.; Wang, D.; Ip, A.; Fung, R.Y.K.(1998). A hybrid genetic algorithm for a type of nonlinear programming problem. *Computers and Mathematics with Application*. 36(5):11-21.
[34] Fung, R.Y.K.;Tang, J.F; Wang, D.W.(2002). Extension of a hybrid genetic algorithm for nonlinear programming problems with equality and inequality constraints. *Computers and operations Research*. 29(3):261-74.
[35] Kelner, V.; Capitanescu, F. (2008). A hybrid optimization technique coupling an evolutionary and a local search algorithm. *Computational and Applied Mathematics*. 215(2): 448-456.
[36] Tang, K.Z.; Yang, J. Y.; Gao, S.; Sun, T.K.(2010). A self adaptive linear evolutionary algorithm for solving constrained optimization problems. *Journal of Control Theory and Application*. Online First™,28 March.
[37] Zhou, Y.R.; Li, Y. X.(2003). A Pareto Strength evolutionary algorithm for constrained optimization. *Journal of Software*, 14(7) : 1243-1249. (In Chinese)
[38] Sarimveis, H.; Nikolakopoulos, A. (2005). A line up evolutionary algorithm for solving nonlinear constrained optimization problems. Computers and Operations Research, 32(6):1499-1514.
[39] Varadarajan, M.; Swarup, K.S. (2008). Differential evolutionary algorithm for optimal reactive power dispatch. Electrical Power and Energy Systems, 30(8): 435-441.
[40] Gegndex, M.E.; Palacios. P.; Lvarez. J.L.(2007). A new self-adaptive crossover operator for real-coded evolutionary algorithms. Lecture Notes in Computer Science, 4431(1):39-48.
[41] Zou, X.F.; Liu.M.Z.(2004). A roubst evolutionary algorithm for constrained multi-objective optimization problems. Computer Research and Development,41(6): 986-990.
[42] Madavan, N.K.(2001). Multiobjective optimization using a Pareto differential evolution approach. Proceedings of 2002 World Congress on Computational Intelligence. Piscataway: IEEE Press, 1145-1150.
[43] Runarason, T.; Yao, X.(2000). Stochastic ranking for constrained evolutionary optimization. IEEE transactions on Evolutionary Computation, 4(3): 284-294.



Evolutionary Algorithms

Edited by Prof. Eisuke Kita

ISBN 978-953-307-171-8
Hard cover, 584 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011

Evolutionary algorithms are successively applied to wide optimization problems in the engineering, marketing, operations research, and social science, such as include scheduling, genetics, material selection, structural design and so on. Apart from mathematical optimization problems, evolutionary algorithms have also been used as an experimental framework within biological evolution and natural selection in the field of artificial life.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen