OPTIMIZATION TECHNIQUES AND AN INTRODUCTION TO GENETIC ALGORITHMS AND SIMULATED ANNEALING

Dr. T. Ghose
Dept. of EEE
BIT, Mesra

INTRODUCTION::

Almost any problem in the design, operation, and analysis of manufacturing plants, and any associated problem can be reduced in the final analysis to the problem of determining the largest and smallest value of a function. So, optimization is the act of obtaining the best result under given circumstances. In most engineering design activities the design objective could be simply to minimize cost of production or to maximize the efficiency of production. For example, optimization required in designing of aircraft and aerospace structures for minimum weight, finding the optimal trajectory of space vehicles, designing of civil engineering structures such as frames, foundations, bridges, towers, chimneys and dams for minimum cost, optimal designing of linkages, cranes, gears, machine tools or pumps, turbines and heat transfer equipment for maximum efficiency, optimal production planning, controlling, and scheduling, optimal operation and control of power system , optimum designing of control system etc.

It is almost impossible to apply a single formulation procedure for all engineering design problems. Since the objective in a design problem and the associated design parameters vary from product to product, different techniques need to be used in different problems. For the reason, it is required to create a mathematical model of the optimal design problem, which then can be solved using an optimization algorithm. The steps involved are, need for optimization, choose design variables ,formulate constraints, formulate objective function, set up variable bounds, choose an optimization algorithm, obtain solution.

It can be stated as follows

$$Find\ X = \begin{Bmatrix} x_1 \\ x_2 \\ , \\ x_n \end{Bmatrix} which\ \min imizes\ f(x)$$

subject to the constrains

$g_j(X) \le 0,\ j = 1, 2, ...m$

$l_j = 0,\ j = 1, 2.....p$

where X is an n-dimensional vector called the design vector, f(X) is termed the objective function, and $g_j(X)$ and lj(X) are inequality & equality constraints respectively.
n = no of variables , m and p = no of constraints

The above problem is called the **constrained optimization problem.** The problems which do not involve any constraint is called **unconstrained optimization problem**

DESIGN VARIABLES::

A design problem usually involves many design parameters, of which some are highly sensitive to proper working of the design. These are called *design or decision variables* x_i, i=1,2,...,n

So design vector, $X=\{x_1, x_2,....,x_n\}$

Certain quantities are usually fixed at the outset & are called *preassigned parameters.*

CONSTRAINTS::

The constraints represent some functional relationships among the design variables & other design parameters satisfying certain physical phenomenon & certain resource limitations. Constraints that represent limitations on the behavior or performance of the systems are termed *behavior or functional constraints.* Constraints that represent physical limitations on design variables such as availability, fabricability, & transportability are known as *geometric or side constraints.*

eg.:- in mechanical & civil Engg. problems , the constraints are formulated to satisfy stress & deflection limitations.

VARIABLE BOUNDS::

There should be some minimum & maximum bounds on each design variables. It is required to confine the search algorithm within these bounds.

e.g. $x_i^{(L)} \leq = x_i \leq = x_i^{(U)}$. If the variables do not lie on the bounded region then chosen bound may be readjusted & the optimization algorithm may be simulated again.

OBJECTIVE FUNCTION::

The criteria with respect to which the design is optimized, when expressed as a function of the design variables, is known as *criterion or merit or objective function.* The objective function for minimization is generally taken as weight in air craft & aerospace structural design problems i.e. in civil engg., minimization of cost. The maximization of mechanical efficiency is the obvious choice of an objective in mechanical design engineering problem. However there may be cases where the optimization w.r.t. a particular constraint may lead to results that may not be satisfactory w.r.t another criteria e.g. in mechanical design a gear box transmitting the maximum power may not have the minimum weight. So selection of objective function is the most important decision. An optimization problem involving multiple objective functions is known as a *multi-objective programming problem.*

If $f_1(x)$ and $f_2(x)$ denote 2 objective functions, then new one will be

$$f(x) = a*f_1(x)+b*f_2(x)$$

where a & b are constants whose values indicate the relative importance of one to the other.

OPTIMIZATION ALGORITHMS::

These are basically divided into two groups
1:-Traditional methods
2:-Non traditional methods

Traditional methods::-These are helpful in finding the optimum solution of continuous & differentiable functions These methods are analytical & make use of the techniques of differential calculus. It provides a good understanding of the properties of the minimum & maximum points in a function & how optimization algorithm work iteratively to find the optimum point in a problem. It is classified into 2 categories

1 .***Direct method***

-Bracketing methods
Exhaustive search method
Bounding phase method
-Region-elimination method
Interval halving method
Fibonasi search method
-Point estimation method
Successive quadratic method

*2.**Gradient method***

-Newton raphson method
-Bisection method
-Secant method
-Cubic search method

Direct method do not use any derivative information of the objective function , only its values are used to guide the search process. Where as Gradient method uses derivative information(1st & 2nd order).

Demerits::

- The convergence to an optimal solution depends on the chosen optimal solution.
- Most algorithms tend to get stuck to a suboptimal solution.
- An algorithm efficient in solving one optimization problem may not be efficient in solving a different optimization problem.
- Algorithms are not efficient in handling problems having discrete variables.
- Algorithms can not be efficiently used on parallel machine.

2- Nontraditional optimization algorithm::

These are quite new methods & are becoming popular day by day. Two such algorithms are

- Genetic Algorithm
- Simulated Annealing

Difference between Genetic algorithm & Traditional optimization algorithm:

-GAs work with a coding of the parameter set not the parameters themselves.
-GAs search from a population of points not a single point.
-GAs use payoff(objective for) information not derivative or other auxiliary knowledge.
-GAs use probabilistic transition rules not deterministic rules.

GENETIC ALGORITHMS

Genetic Algorithm is a nontraditional and optimization method based on the mechanics of natural genetics and natural selection. Professor John Holland of the University of Michigan envisaged the concept of these algorithms in the mid sixties. Thereafter a number of his students and other researchers have contributed to developing this field. GAs are now becoming very much popular in engineering optimization problems for his wide range of precise search and capability of solving complex non-linear problems. Research works are going on to extend the efficiency of Genetic Algorithms as well as the implementation technique of GAs on various problems. The different steps of GAs are now outlined in the working principle.

Working Principles:

Unlike many mathods, GAs use probabilistic transition rules to guide their search. The method is not a simple random search or is not a decision making tool depending on the simple probability act just like a toss of a coin. GAs use random choice as a tool to guide a search toward regions of the search space with likely improvement.

To demonastrate the working principles of GAs, the following maximization problem is considered

$$\textit{Maximize} \quad \mathrm{f}(x),\ x_{\mathrm{i}}^{(L)} \le x_i \le \mathrm{x}_{\mathrm{i}}^{(U)},\ i = 1,2,.....\mathrm{N}$$

Although a maximization problem is considered here, a minimization problem can also be handled using GAs. The working of GAs is completed by performing the following tasks:

Coding: To implement GAs in the solution of the above maximization problem, variable x_i's are first coded in some string structures. Variable x_i's are coded by binary

representation having 0's and 1's. The length of the coded string is usually determined according to the desired solution accuracy. For example, if four bits are used to code each variable in a two variable function optimization problem, the strings (0000 , 0000) and (1111, 1111) would represent the points $(x_1^{(L)} , x_2^{(L)})^T$ $(x_1^{(U)}, x_2^{(U)})^T$, respectively, because the substring (0000) and (1111) have the minimum and miximum decoded values. Any other eight bit string can be found to represent a point in the search space according to a fixed mapping rule. Usually, the following linear mapping rule is used:

$$x_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^{Li} - 1} \text{ decoded value}(S_i)$$

In the above equation, the variable x_i is coded in a substring S_i of length l_i. The decoded value of a binary substring S_i is calculated as

$\sum_{i=0}^{l-1} 2^i S_i$, where $S \in (0,1)$ and the string S is represented as (S_{l-1} S_{l-2}.... $S_2 S_1 S_o$). For example, a four bit string (0 111) has a decoded value equal to ((1) 2^o + (1) 2' + (1) 2^2 + (0) 2^3) or 7. It is worthwhile to mention here that with four bits to code each variable, there are only 2^4 or 16 distinct substrings possible, because each bit position can take a value either 0 or 1. The accuracy that can be obtained with a four bit coding is only approximately 1/16th of the search space. But as the string length is increased by one, the obtainable increases exponentially to 1/32th of the search space.

Initialization :

Referring to the maximization problem a set of binary strings representing the variable x_i are generated at random to make the initial population. The string in GA correponds to ' "chromosome" and bits in a string refers "genes" in natural genetics.

Fitness Function :
Every member string in a population is judged by the functional value of the fitness function. As GAs follow the rule of survival-of-the-fittest candidate in nature to make a search process so the algorithm is naturally suitable for solving maximization problems. Minimization problems are ususally transformed into minimization problems by some suitable tranformation. In general, a fitness function F(x) is first derived from the objective function and used in successive genetic operations.

Genetic Operators:

With an initial population of individuals of various fitness values, the operators of GAs begin to generate a new and improved population from the old one. A simple genetic

algorithm (SGA) consists of three basic operations : reproduction, crossover and mutation. Through these operations a new population of points is evaluated. The population is iteratively operated by the above three operators and evaluated until the goal or termination criterion is met. One cycle of these operations and subsequent evaluation procedure is known as generation in GAs.

Reproduction:

Reproduction is usually the first operator applied on a population. Reproduction selects strings according to the fitness values in a population and forms a mating pool. Selecting strings according to their fitness values means that string with a higher value have a higher probability of contributing one or more off springs to the next generation. The i-th string in the population is selected with a probability proportional to F_i. Since the population size is usually kept fixed in a simple GA, the sum of the probability of each string being selected for the mating pool must be one. Therefore, the probability for selecting the i-th string is

$$\rho_i = \frac{F_i}{\sum_{j=1}^{n} F_j},$$

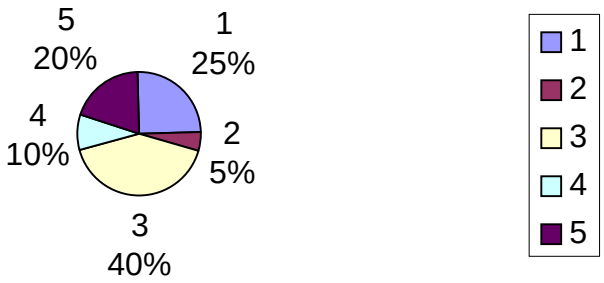
where, n is the population size. One way to implement this selection scheme is to imagine a roulette-wheel with its circumference marked for each string proportionate to the string's fitness. The roulette-wheel is spun n times, each time selecting an instance of the string chosen by the roulette-wheel pointer. Since the circumference of the wheel is marked according to a string's fitness, the roulette-wheel mechanism is expected to make $F_i/\bar{F}$ copies of the i-th string in the mating pool. The average fitness of the population is calculated as,

$$\bar{F} = \sum_{i=1}^{n} F_i \Big/ n$$

Fig. 1 shows a roulette-wheel for fine individuals having different fitness values. Since the third individual has a higher fitness value than any other, it is expected that the roulette-wheel selection will choose the third individual more than any other individual. This roulette-wheel selection scheme can be simulated easily.

Using the fitness value F_i of all strings, the probability of selecting a string P_i can be calculated. Therefore, the cumulative probability (P_i) of each string being copied can be calculated by adding the individual probabilities from the top of the list.

Fig.1: A roulette -wheel marked for five individuals according to their fitness values.



Point	Fitness
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

Crossover:

In reproduction, good strings in a population are probabilistically assigned a larger number of copies and a mating pool is formed. But no new strings are formed in the reproduction phase. In the crossover operator, new strings are created by exchanging information among strings of the mating pool. Many crossover operators exists in the GA literature. In most crossover operators, two strings are picked from the mating pool at random and some portions of the strings are exchanged between the strings. A single - point crossover operator is performed by randomly choosing a crossing site along the string and by exchanging all bits on the right side of the crossing site as shown:

0 0 0 | 1 0 0 　　 0 0 0 | 1 0 1

⇒

1 1 | 1 0 1 　　 1 1 1 | 1 0 0

The two strings participating in the crossover operation are known as parent strings and the resulting strings are known as children strings. It can be expected that good substrings from parent strings can be combined to form a better child string, if an appropriate site is chosen. Since the knowledge of an appropriate site is usually not known beforehand, a random site is often chosen. With a random site, the children strings produced may or may not have a combination of good substrings from parent strings, depending on the position of crossover point. If good children are not produced from crossover operator, there is no need to worry so much about this. Because reproduction operator will select these strings with fewer copy in subsequent strings as a result they will not survive too

long. It is clear from this discussion that the effect of crossover may be detrimental or beneficial. Thus, in order to preserve some of the good strings that are already present in the mating pool, not all strings in the mating pool are used in crossover.

*Mutation **:***
A crossover operator is mainly responsible for the search of new strings, even though a mutation operator is also used for this purpose. The mutation operator changes 1 to 0 and vice versa in a bit position with a small mutation probability, p_m. Changing bit with probability p_m can be simulated by choosing a number between 0 to 1 at random. If the random number is smaller than p_m, the randomly selected bit is altered; otherwise the bit is kept unchanged. The need for mutation is to create a point in the neighbourhood of the current point, thereby achieving a local search around the current solution. The mutation is also used to maintain diversity in the population. For example, consider the following population having four eight-bit strings:

0 1 1 0 1 1 0 0
0 0 1 0 0 0 1 1
0 1 0 1 1 1 1 1
0 1 1 1 0 0 0 0

Notice that all four strings have a 0 in the left-most position. If the true optimum solution requires 1 in that position, then neither reproduction nor crossover operator described above will be able to create 1 in that position. The inclusion of mutation introduces some probability of turning 0 into 1.

These three operators are simple and straightforward. A number of research papers have so far been conducted to improve the efficiency of GAs. Some variations have been introduced in GAs operators. In most cases, the variants are developed to suit particular problems.

Advantages of GAs :

As seen from the above description of the working principles of GAs, they are radically different from most of the traditional optimization methods. However, the general advantages are described in the following paragraphs.
GAs work with a string-coding of variables instead of the variables. The advantage of working with a coding of variables is that the coding discretizes the search space, even though the function may be continuous. On the other hand, since GAs require only function values at various discrete points a discrete or discontinuous function can be handled with no extra cost. This allows GAs to be applied to a wide variety of problems. Another advantage is that the GA operators exploit the similarities in string-structures to make an effective search.

The most striking difference between GAs work with a population of points instead of a single point. Because there are more than one string being processed simultaneously, it is very likely that the expected GA solution may be a global solution. Even though some traditional algorithms are population-based, like Box's evolutionary optimization and complex search methods, those methods do not use previously obtained information efficiently. In GAs, previously found good information is emphasized using reproduction operator and propagated adaptively through crossover and mutation operators. Another advantage with a population-based search algorithm is that multiple optimal solutions can be captured in the population easily, thereby reducing the effort to use the same algorithm many times.

In discussing GA operators or their working principles in the previous section, nothing has been mentioned about the gradient or any other auxiliary problem information. In fact, GAs do not require any auxiliary information except the objective function values. Although the direct search methods used in traditional optimization methods do not explicitly require the gradient information, some of those methods use search directions that are similar in concept to the gradient of the function. Moreover, some direct search methods work under the assumption that the function to be optimized is unimodal and continuous. In GAs, no such assumption is necessary.

One other difference in the operation of GAs is the use of probabilities in their operators. None of the genetic operators work deterministically. The basic problem with most of the traditional methods is that they use fixed transition rules to move from one point to another. For instance, in the steepest descent method, the search direction is always calculated as the negative of the gradient at any point, because in that direction the reduction in the function value is maximum. In trying to solve a multimodal problem with many local optimum points, search procedures may easily get trapped in one of the local optimum points. But in GAs, an initial random population is used, to start with, the search can proceed in any direction and no major decisions are made in the beginning. Later on, when the population begins to converge in some bit positions, the search direction narrows and a near optimal solution is achieved. This nature of narrowing the search space as the search progresses is adaptive and is a unique characteristic of genetic algorithms.

An Examples

Design of simple can by GA

A cylindrical can is considered to have only two design parameters- diameter d and height h. Let us consider that the can needs to have a volume of at least 300ml and the objective of the design is to minimize the cost of can material.

Nonlinear objective function

$$f(d,h)) = c((\pi d^2/2 + \pi dh),$$
$$Subjecttov(d,h) = (\pi d^2/4) \geq 300$$
$$\text{var}\, iablebounds\, \mathrm{d}_{\min} \leq d \leq d_{\max}$$
$$\mathrm{h}_{\min} \leq h \leq h_{\max}$$

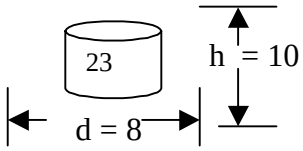
Step 1

In order to get optimal parameter values of d and h which satisfy the constraint g_4 and minimize f we first need to code the parameter values in binary strings.

Assume five bits are representing each of the parameters. So the overall string length is equal to 10.

The following string represents a can of diameter 8 cm and height 10 cm.

01000 01010
d h



Step 2

In the above representation, the lower and upper values of both parameters are considered to be zero and 31. So mapping function is not required to use to get the exact value as minimum (00000) and maximum value (11111) represent 0 and 31 respectively which satisfy the lower and upper bounds considered for d and h parameters. But GAs can be assigned to use any integer or non-integer values just by changing the string length, lower and upper bounds.

$$x_i = x_i^{\min} + \frac{x_i^{\max} - x_i^{\min}}{2^L - 1} \times \text{Decoded value of string i.}$$

In the above example L = 5, $x_i^{\min} = 0$, $x_i^{\max} = 31$

Step 3

Assigning fitness to a solution

The fitness is made equal to the objective function value. For example, the fitness of above can

$$F(S) = 0.0654 \times \left(\frac{\Pi 8^2}{2} + \Pi\,(8)\,(1\ 0) \right)$$

= 23 (assuming C = .0654)

Since the objective of the optimization is to maximize the objective function, it is to be noted that a solution with a smaller fitness value is better compared to another solution.

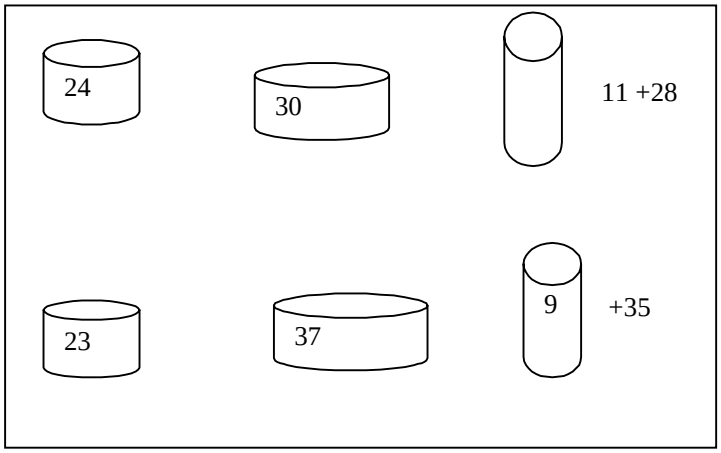


Fig. 2 A random population of 6 cans.

Fig. 2 shows a random population of 6 cans. This fitness (penalized cost) of each can is marked on the can. It is interesting to note that two solutions do not have 300 ml. volume inside and thus have been penalized by adding an extra artificial cost thus making the infeasible ones become worse solutions.

Step 4

Reproduction operator : The primary objective of the reproduction operator is to emphasize good solutions and eliminate bad solutions in a population, while keeping the population size constant. This is achieved by performing the following tasks:

(1) Identifying good (usually above-average) solution in a population .
(2) Making multiple-copies of good solutions.
(3) Eliminating bad solutions from the population so that multiple copies of good solutions can be placed in the population.

There exist a number of ways to achieve the above tasks. Some common methods are tournament selection- proportionate selection, ranking selection and others. In the following, we illustrate the binary tournament selection.

As the name suggests, tournaments are played between two solutions and the better solution is chosen and placed in a population slot. Two other solutions are picked again and another population slot is filled up with the better solution. If done systematically, each solution can be made to participate in exactly two tournaments. The best solution in a population will win both times, thereby making two copies of it in the new population. Similarly, the worst solution will lose in both tournaments and will be eliminated from

the population. This way, any solution in a population will have zero, one, or two copies in the new population.

Figure 3 shows the six different tournaments played between the old population members (each gets exactly two turns) shown in figure 3. When cans with a cost of 23 units and 30 units are chosen at random for tournament, the can costing 23 units is chosen and placed in the new population . Both cans are replaced in the old population and two cans are chosen for other tournaments in the next round. This is how the mating pool is formed and the new population after reproduction shown in box of fig . 3 is created. It is interesting to note how better solutions (having lesser costs) have made themselves have more than one copy in the new population and worse solutions have been eliminated from the population. This is precisely the purpose of a reproduction operator.

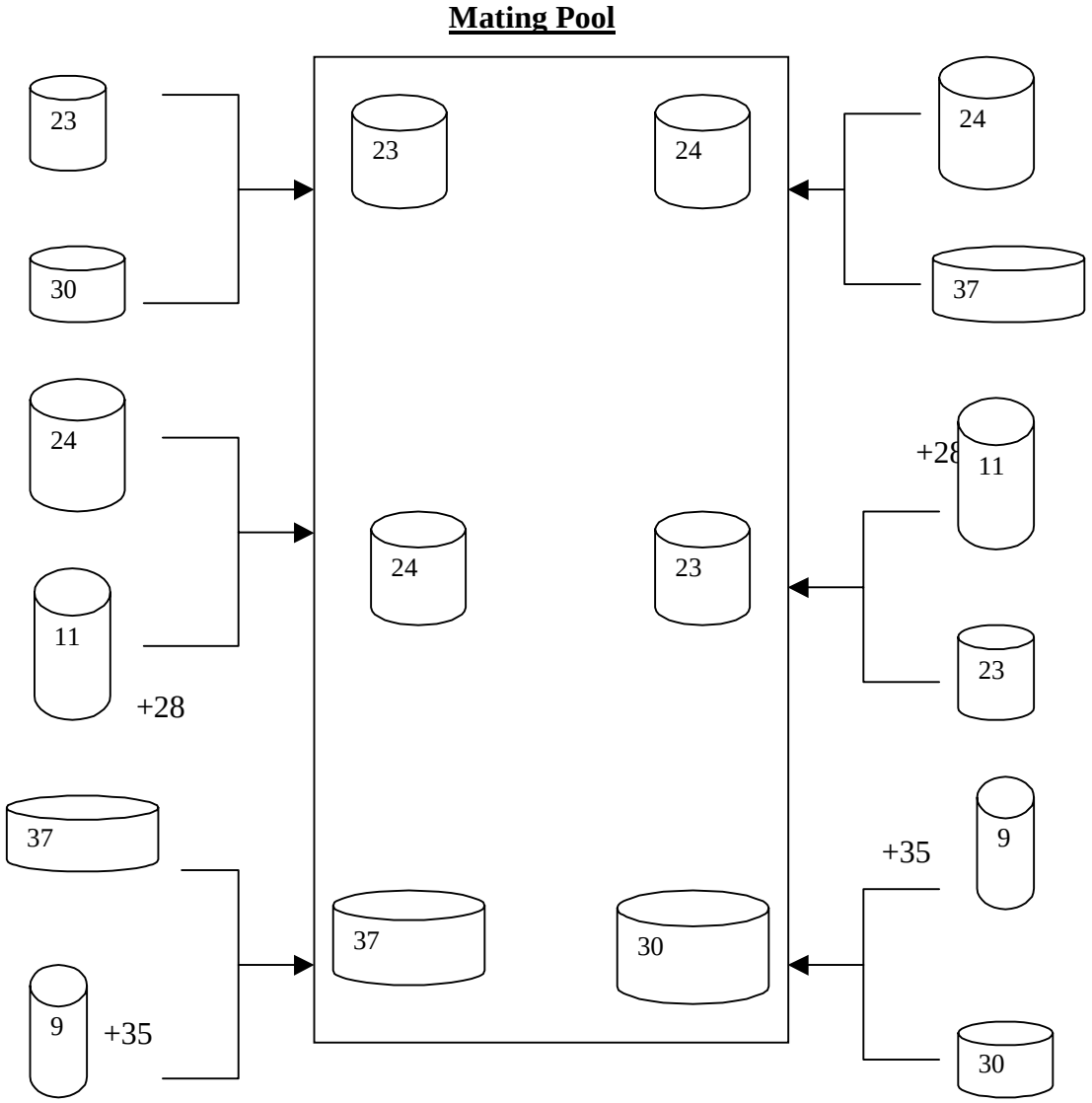


Figure 3. Tournaments played between six population members are shown. Solutions within the dashed box form the mating pool.

Step 5

Like the reproduction operator, there exist a number of methods for crossover operations. Here a single point crossover operation is illustrating for the CAN problem. In fig. 4, reproduction operator selects two strings and the third site along the string length in chosen at random and contents of the right side of this cross site are swapped between the two strings. The process creates two new strings.

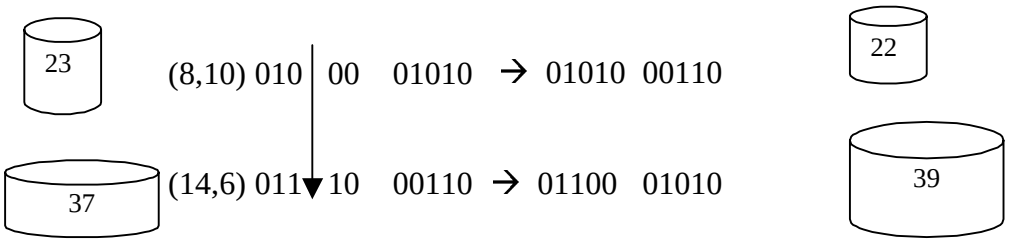


Fig. 4 : An illustration of the single point crossover operator.

In order to preserve some good strings selected during the reproduction operator, not all strings in the population are used in crossover. If a crossover probability of P_c is used then 100 P_c% strings in the population are used in the crossover operation and (100 (1-Pc)% of the population are simply copied to the new population.

Step 6

The mutation operator changes 1 to a 0 and vice versa with a small mutation probability P_m. The need for mutation is to keep diversity in the population. Fig. 5 shows how a string obtained after reproduction and crossover operator has been mutated to another string representing a slightly different CAN.

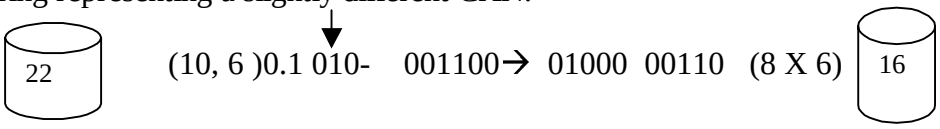


Fig. 5 An illustration of the mutation operation.

Constrained optimization using GA

An optimal design problem having N variables is written as a nonlinear programming

(NLP) problem as follows:

Minimize f(x)

$$Subject to\, g_j(x) \geq 0 \; j = 1,2.....J.$$

$$h_k(x) = 0 \quad k = 1,2.....K$$

$$x_i^{\min} \leq x_i \leq x_i^{\max} i = 1,2...$$

In the above problem there are J inequality and K equality constraints. The can design problem has two (N=2) variables one (J=1) inequality constraint and no (K = 0) equality constraint. The simple penalty function method converts the above constrained NLP problem to an unconstraint minimization problem by penalizing infeasible solutions:

$$P(x,R,r) = f(x) + \sum_{j=1}^{J} R_j (g_j(x))^2 + \sum_{k=1}^{K} r_k (h_k(x))^2$$

The parameters R_j and r_k are the penalty parameters for inequality and equality constraints respectively.

Modified GA:

Some phenomena in natural genetic system are emulated in fitness function evaluation and crossover operation in order to improve the efficiency of conventional genetic algorithms [4]. These include the concept of aging of individuals and ancestors' influence for computing the fitness value of individuals, and genotypic and phenotypic similarity for determining pairs undergoing crossover operation. Two methods have been discussed in below:

Aging of individuals

In the conventional GA once a particular solution becomes more fit, it goes on getting chances to produce offspring until the end of the algorithm, if a proportional payoff (of the objective functional value only) selection is used; thereby increasing the chance of generating similar type of offspring and loosing diversity very fast. More fit individuals do not normally die (i.e., they are not deleted from the population), and only the less fit ones die. Thus in order to maintain diversity, larger population size is needed; and this in turn increases the computational burden and slows down the convergence process. In the present work, fitness of each individual with respect to age is assigned in such a, way that after a pre-defined upper age limit (number of iterations), this value becomes zero. This, more or less, ensures a natural death (deletion from the population) for each individual keeping its offspring only alive. Thus, in this case a particular individual cannot dominate for a longer period of time. This helps to maintain diversity in the population even with smaller population size. It can be mentioned here that, at the onset when the population is initialized, age of individuals may be initialized either to one or to any positive value with an upper limit.

Incorporation of ancestor's influence

As is well known in GAs, in each generation or iteration, the objective function (fitness measuring criterion) determines the suitability of each individual. Based on these, some of them, called parent individuals, are selected for reproduction. Number of copies reproduced by an individual parent is expected to be directly proportional to its fitness value; hence, the performance of a GA depends on the fitness evaluation criterion, to a large extent. Genetic operators are applied on these (selected) parent individuals and new individuals (offspring) are generated. Conventional genetic algorithms (CGAs) consider only the fitness value of the individual under consideration for measuring its suitability for selection for the next generation i.e., the fitness of an individual I_i is $fit_i = g(fv_i)$, where fv_i is the objective function and g is another function which by operating on fv_i gives the fitness value. Hence, a CGA does not discriminate between two identical offspring, one coming from better (highly fit) parents and the other from comparatively weaker (low fit) parents. In nature, normally an offspring is found to be more it (suitable) if its ancestors (parents) are well off i.e., an offspring possess some extra facility to exist in its environment if it belongs to a better family (ancestors are highly fit). In other words, the fitness of an individual in Gas should depend also on the fitness of its ancestors in addition to its own fitness.

Based on this realization, we describe in this section a concept for measuring the fitness of an individual by considering its own fitness as well as that of its ancestors' i.e., fitness of an individual Ii is $fit_i = g(fv_i, a_1, a_2, \ldots\ldots, a_n)$ where a_is are the fitness values of its ancestors [12]. The function g may be of various types considering the amount of importance to be given on the fitness of different ancestors. The weighting factors may be kept constant or varying during the operation of GAS.

Simulated Annealing

Simulated Annealing (SA) is another nontraditional search and optimization method which is becoming popular in engineering optimization problems. For the solution of combinatorial optimization problem, SA is very much suitable and powerful search technique.

The simulated annealing method resembles the cooling process of molten metals through annealing. At high temperature, the atoms in the molten metal can move freely with respect to each another, but as the temperature is reduced, the movement of the atoms gets restricted. The atoms start to get ordered and finally form crystals having the minimum possible energy. However, the formation of the crystal mostly depends on the cooling rate. If the temperature is reduced at a very fast rate, the crystalline state may not be achieved at all, instead, the system may end up in a polycrystalline state, which may have a higher energy state than the crystalline state. Therefore, in order to achieve the absolute minimum energy state, the temperature needs to be reduced at a slow rate. The process of slow cooling is known as annealing in metallurgical parlance.

Simulated annealing is a point-by-point method. The algorithm begins with an initial point and a high temperature T. A second point is created at random in the vicinity of the initial point and the difference in the function values (ΔE) at these two points is calculated. If the second point has a smaller function value, the point is accepted; otherwise the point is accepted with a probability exp (− Δ E/T). This completes one iteration of the simulated annealing procedure. In the next generation, another point is created at random in the neighbourhood of the current point and the metropolis algorithm is used to accept or reject the point. In order to simulate the thermal equilibrium at every temperature, a number of points (n) is usually tested at a particular temperature, before reducing the temperature. The algorithm is terminated when a sufficiently small temperature is obtained or a small enough change in function values is found.

The design of an algorithm based on simulated annealing consists of four important elements: (1) a set of allowed system configurations (configuration space), (2) a set of feasible moves (move set), (3) a cost function and (4) a cooling schedule. The system to be optimized starts at a high temperature and is slowly cooled down, until the system freezes and reaches the global optimum in a manner similar to annealing of a crystal during growth to reach a perfect structure. At each temperature, the simulated annealing algorithm is represented in the following pseudo-code

repeat

1. perturb

2. evaluate C

3. accept/update

until stop criterion = true

Step-1 perturb the current system configuration to a new configuration. Step-2 evaluate the change of the cost function dc = c -c. where c and c are the value of the cost function before and after the move has been executed. Step-3, accept/update if the move decreases the value of the cost function i.e. dc < 0, the move is accepted and the new configuration is retained. On the other hand when dc > 0 (i.e. the move is uphill) acceptance is treated probabilistically in the following way the Boltzman factor $e^{-\frac{dc}{kT}}$ is first calculated, where the parameter T is the temperature and k is a constant whose dimension depends on c and T. Then a random number r uniformly distributed in the interval [0,1] is chosen.

If $r \leq e^{-\frac{dc}{kT}}$, the new configuration is retained; otherwise, if $r \geq e^{-\frac{dc}{kT}}$, the move is discarded and the configuration before this move is used for the next step. The algorithm stops when no significant improvement in the cost function has been found for a number of consecutive iterations.

The quality of the final solution and the speed of convergence of algorithms based on simulated annealing depend on the choices of k and the initial temperature in conjunction with the design of the cooling schedule. The temperature is initially high value so that the probability of accepting uphill moves are close to 1, and then it is slowly decreased towards frozen according to a cooling schedule. If the annealing is slow enough, the probability that the optimal system configuration will be achieved is close to 1. Due to the probabilistic selection rule, the process can always get out of a local minimum in which it could get trapped and proceed to the desired global optimum. This feature makes the simulated annealing different from the greedy search approach.

On the basic of the above analogy, the overall algorithm of the simulated annealing method described below can be derived.

Step 0: Initialization

Initialize the iteration count k = 0, and the temperature T^0 to be sufficiently high. Find an initial solution x^0.

Step 1: Repeat for each temperature T^k

Execute Steps 2-4 until an equilibrium criterion is satisfied.

Step 2: Neighborhood solution

Generate a trial solution x^{k+1} in the neighborhood of the current solution x^k.

Step 3: Acceptance criterion

Let $\Delta = f(x^{k+1}) - f(x^k)$

And r is a random number uniformly distributed over [0,1]. If $\Delta < 0$ (i.e., the solution is improved), the trial solution is accepted. Otherwise, the trial solution is accepted with the probability

$\exp(-\Delta/T^k) > r$

Step 4: Cooling schedule

Gradually decrease the value of the temperate T^k by

$T^{k+1} = p * T^k$, $0<p<1$

Steps 5: Convergence check

It the number of the accepted solution is small enough, freezing point is reached and the algorithm is terminated. Otherwise, set k=k+1 and go to Step 1.

The flow chart of the simulated annealing method above is shown in Fig given below.

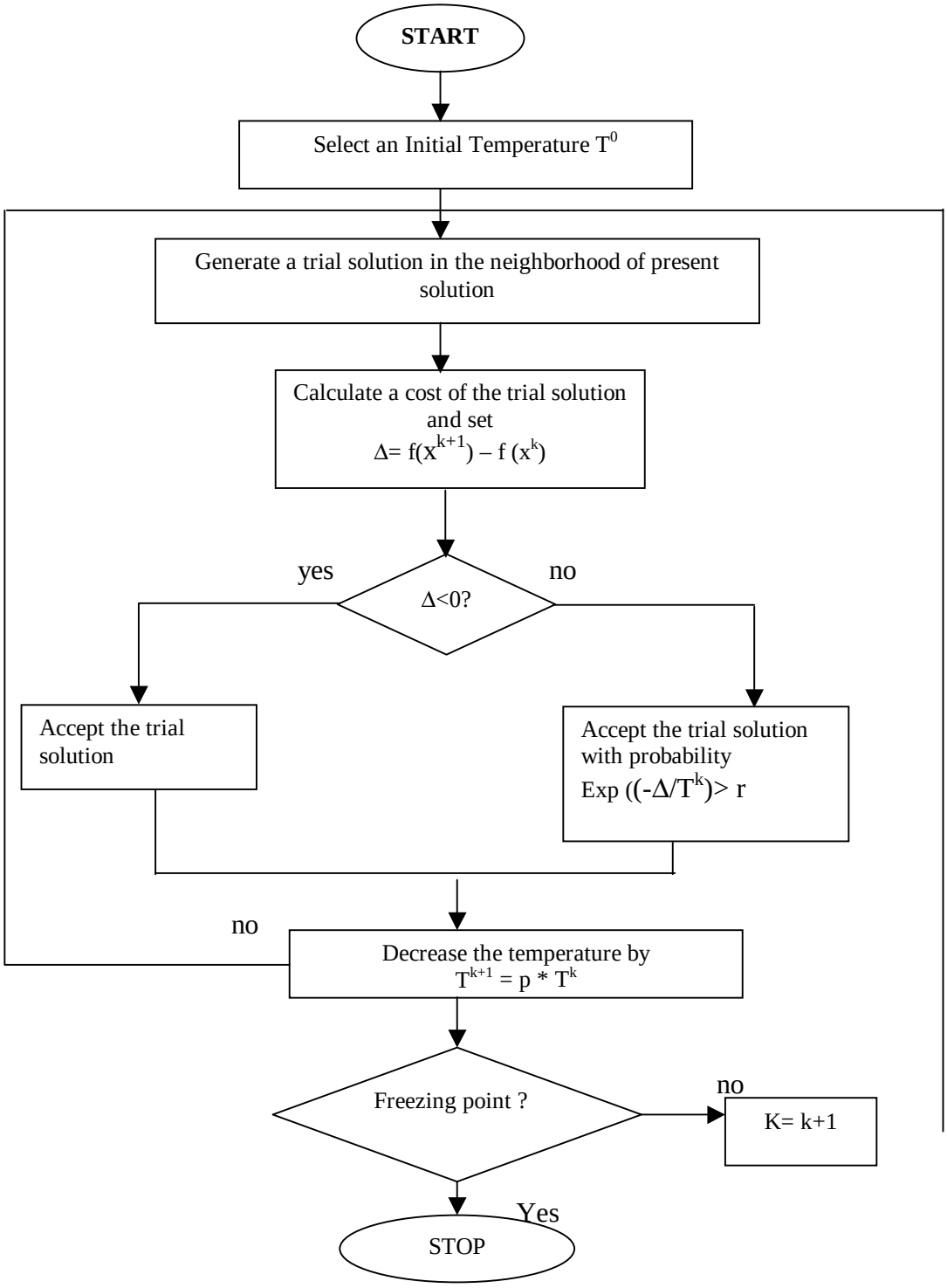


Fig. 1: Flow chart of simulated annealing

References

1. Goldberg, D.E. " Genetic Algorithms in search, optimization, and machine learning" Addition-Wesley

2. K. Deb, "Optimization for Engineering Design" PHI, New Delhi.

3. T. Ghose, Dr. S. K. Goswami & Professor S. K. Basu, " Solving Capacitor Placement Problem In Distribution Systems Using Genetic Algorithms " , *Published in Vol. 27, Number 4,1999 of international jounal of Electric Machines and Power Systems.*

4. Ashish Ghosh and S.K. Pal,"Biologically Inspired New Operations for Genetic Algorithms" Published in prceeding of int. workshop on soft computing and Int. Systems,ISI Calcutta, January 12-13, 1998.