Electric Power Systems Research, *22* (1991) 151 - 163

Investigations on Solving the Load Flow Problem by Genetic Algorithms

XIAODONG YIN and NOËL GERMAY

Laboratoire d'Electrotechnique et d'Instrumentation, Catholic University of Louvain, Place de Levant 3, 1348 Louvain-La-Neuve (Belgium)

(Received June 19, 1991)

ABSTRACT

This paper presents the application of genetic algorithms—search procedures based on the mechanics of natural genetics and natural selection—to the load flow problem in electrical power systems. Numerical experiments on small-size networks show that the simple genetic algorithm (SGA) quickly finds an abnormal solution. By specifying an additional term in the objective function, the normal load flow solution can be found. An attempt to search for multiple load flow solutions simultaneously with the help of a sharing function is made and examined. Finally, an approach to improve the precision of genetic algorithm results is presented.

1. INTRODUCTION

The load flow problem is one of the most important problems in power system planning, operation and control. Load flow is the solution for the static operating condition of an electric power system. Since the load flow equations are algebraic nonlinear, many numerical methods [1] have been developed for finding the desired normal solution. Among these, the Gauss–Seidel method using the nodal admittance matrix, which requires minimal computer storage, and the Newton–Raphson method, which is reliable and widely accepted when ordering and sparsity techniques are used to solve the large-scale network problem. Some variations of the Newton–Raphson method have been proposed recently, amongst which the fast decoupled method is the best known. It has gained wide acceptance in the power industry due to its simplicity and computational efficiency. All the above methods need an initial guess value to start. A careless or random selection of initial values may cause the methods to miss the normal solution, either by divergence or by convergence to an abnormal solution [2].

On the other hand, owing to the nonlinearity of the load flow equations, the possibilities of no solution, a unique solution or multiple solutions may exist [3]. The *multiple load flow solution* problem arises because, when a set of nonlinear equations is solved, there may be many solutions which satisfy the same specified conditions. It has been pointed out [4] that multiple solutions can be realized physically, and several examples in a practical system have been given [5]. Recently, it was observed that the voltage instability phenomena that tend to occur under heavily loaded conditions seem to be related to the multiple load flow solution problem [6]. In ref. 7, the load flow equations were analyzed theoretically by the methods of algebraic geometry. Under the assumptions of a lossless network and fixed bus voltage magnitudes, it was concluded that the number of (real) solutions of the load flow equations for an n-node network is bounded below by 2^{n-1} and above by C_{2n-2}^{n-1} and depends on the network parameters. Few studies have been devoted to find the multiple load flow solutions. The method proposed in ref. 8 uses a technique to generate a set of adequate initial values systematically, then repeats the traditional Newton–Raphson method to find the multiple solutions successively. However, the initial values are not always located in the convergence regions, so that the method may fail to find some solutions. More recently, a method for finding a pair of multiple solutions that are located closely to each other was proposed [9]. There still remain fields to explore in the search of methodologies for determining the nature of the multiple load flow solutions as well as for finding them numerically.

0378-7796/91/$3.50

© Elsevier Sequoia/Printed in The Netherlands

In this paper, a series of investigations is carried out to solve these problems by means of genetic algorithms (GAs). Originally inspired by biological genetics, and then adapted by Holland [10], genetic algorithms are a kind of stochastic search algorithm based on the mechanics of natural selection and natural genetics [11]. They combine an artificial survival of the fittest with genetic operators abstracted from nature to form a surprisingly robust mechanism that is suitable for a variety of search problems. Genetic algorithms have been applied in various areas [12 - 19] such as pipeline engineering, VLSI microchip layout, communication network design, inverse kinematics of redundant robots, parameter estimation, keyboard configuration design, medical image registration, and structural optimization problems. In this paper, the structure and properties of GAs are described. Then, the load flow problem is outlined. The implementation of a simple genetic algorithm and its application to a Klos–Kerner three-node system and a Ward–Hale six-node system are presented and examined. By incorporating an additional term in the objective function, the normal solution is found. Next, the multiple load flow solution problem is investigated, using the concepts of niche and sharing. Finally, an approach is tried to improve the GA results.

2. OVERVIEW OF GENETIC ALGORITHMS

Genetic algorithms are essentially derived from a simple model of population genetics. They have five components as follows [20]:

(1) chromosomal representation of the variables characterizing an individual;

(2) an initial population of individuals;

(3) an evaluation function that plays the role of the environment, rating the individuals in terms of their 'fitness', that is, their aptitude to survive;

(4) genetic operators that determine the composition of a new population generated from the previous one by a mechanism similar to sexual reproduction;

(5) values for the parameters that the GAs use.

2.1. Representation

Because GAs are based on natural genetics, there exist strong analogies between genetic algorithms and natural genetics. In GAs, the *features* characterizing an individual are often binary coded in *bits* (e.g. 0s and 1s) and concatenated as a *string*. The strings are similar to *chromosomes* in biological systems, where the chromosomes are composed of *genes* which may take any of several forms called *alleles*. In natural systems, the total genetic package which may be created from different combinations of alleles is called a *genotype* and the totality of the observed behavior of a genotype is called a *phenotype*. In artificial genetic systems, the string package made from different combinations of bits is named a *structure*; its decoded interpretation is referred to as a *variable alternative*, or a point in the solution space.

2.2. Initialization

GAs do not work with a single string but with a population of strings, which evolves iteratively by generating new individuals taking the place of their parents. To begin, the initial population is generated at random or through the use of specified information.

2.3. Evaluation function

The performance of each structure is evaluated according to its 'fitness', which is defined as the non-negative figure of merit to be maximized. It is associated directly with the objective function value g in optimization. GAs treat the problem as a 'black box' in which the input is the structure of the string and the output is its fitness f. Because GAs proceed only according to the fitness of the strings and not to other information, the properties of fitness will influence the GA's performance. Firstly, the performance of GAs is highly sensitive to the fitness normalization. Secondly, when GAs are applied as optimizers, it often happens that different applications have different requirements. This can be done by incorporating these requirements into the evaluation function without altering the genetic algorithm and genetic operators at all.

GAs are essentially unconstrained search procedures within the given representation space. The constraints can be implemented in the evaluation function in the form of penalty terms or used directly as specified restrictions that avoid creating individuals that violate the constraints.

2.4. Genetic operators

With an initial population of individuals of various fitnesses, 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.

2.4.1. Reproduction (selection)

Reproduction is simply an operation whereby an old string is copied into a 'mating pool' according to its fitness. More highly fitted strings (i.e. with better values of the objective function) receive a higher number of copies in the next generation. This principle is, in fact, an artificial version of the Darwinian survival of the fittest theory in populations of creatures.

2.4.2. Crossover

Crossover is a randomized yet structured recombination operation. Simple crossover may proceed in two steps. Firstly, the newly reproduced strings in the mating pool are mated at random. Secondly, each pair of strings crosses over as follows: an integer position p along the string is selected uniformly at random in the intervals $[1, L-1]$, where L is the string length; two new strings are created by swapping all characters between position 1 and p inclusively. To see how this works, consider two strings A and B of length 10 mated at random from the mating pool created by previous reproduction:

$$A = a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10}$$

$$B = b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8 \quad b_9 \quad b_{10}$$

Suppose the roll of a die turns up a 6, the resulting crossover yields two new strings A′ and B′ following the partial exchange:

$$A' = b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad a_7 \quad a_8 \quad a_9 \quad a_{10}$$

$$B' = a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad b_7 \quad b_8 \quad b_9 \quad b_{10}$$

Crossover exchanges corresponding genetic material from two parent chromosomes, allowing beneficial genes on different parents to be combined in their offsprings.

2.4.3. Mutation

Until now, reproduction and crossover effectively search and recombine existing chromosomes; they do not create any new genetic material in the population. Mutation is capable of overcoming this shortcoming. It is an occasional random alteration of a string position. In the binary string representation, this simply means changing a 1 to a 0 or vice versa. This random mutation provides background variation and occasionally introduces beneficial materials into the population.

2.5. Parameter values

Like other stochastic methods, GAs have a number of parameters (population size, probabilities of genetic operations) that must be selected. Usually, relatively small population size, high crossover probabilities, and low mutation probabilities inversely proportional to the population size are recommended.

Since GAs work on the feature or variable codes rather than on the variable values, they are specially robust and indifferent to the problem specificities like the values of the derivatives or their continuity. GAs work with a population of strings, climbing many peaks in parallel, so they are highly efficient for searching in multidimensional spaces with multimodal functions. Holland [10] has shown through his schemata theory that GAs work in an implicitly parallel way. Schemata are substring bit configurations transmitted to the new population by the genetic operators, and having their own fitness, defined in a statistical way. Holland has demonstrated the ability of these operators to propagate exponentially and in parallel the best schemata through the population, generation after generation. It is this *implicit parallelism* that results in the processing rapidity of GAs. Moreover, although most often studied on classical Von Neumann computer architectures, the similar treatment of all the strings of a population is ideally suited to a parallel processor architecture. Simulation and hardware realization on different parallel architectures was begun in the middle 1980s [11].

Our motivation for applying GAs to the load flow problem is based on these features. GAs need not calculate the Jacobian matrix which relates power mismatch to voltage increment, as they only require the load flow equations and a fitness function. Their capacity for solving multimodal functions in a parallel way prompted us to use them to investigate the multiple load flow solution problem.

3. LOAD FLOW PROBLEM

The load flow equations can be developed either in polar coordinates or in rectangular coordinates. We have used both but present here the formulation in rectangular coordinates and the variables in per unit.

Consider an interconnected n-node power system where there are n_{PQ} load nodes, n_{PV} voltage-controlled nodes and one slack node. In rectangular coordinates, there are $2(n-1)$ unknowns to solve. The load flow equations are:

$$P_i^{sp} = E_i \sum_{j=1}^{n} (G_{ij}E_j - B_{ij}F_j) + F_i \sum_{j=1}^{n} (G_{ij}F_j + B_{ij}E_j) \qquad i \in n_{PQ} + n_{PV} \tag{1}$$

$$Q_i^{sp} = F_i \sum_{j=1}^{n} (G_{ij}E_j - B_{ij}F_j) - E_i \sum_{j=1}^{n} (G_{ij}F_j + B_{ij}E_j) \qquad i \in n_{PQ} \tag{2}$$

$$(V_i^{sp})^2 = E_i^2 + F_i^2 \qquad i \in n_{PV} \tag{3}$$

where

$$\bar{V}_i = E_i + jF_i; \quad Y_{ij} = G_{ij} + jB_{ij}$$

The objective function results from the summation of squares of the power mismatch and the voltage mismatch whose minimum (ideally 0) coincides with the load flow solution:

$$\min g(E, F) = \sum_{i \in n_{PQ}+n_{PV}} \Delta P_i^2 + \sum_{i \in n_{PQ}} \Delta Q_i^2 + \sum_{i \in n_{PV}} \Delta V_i^2 \tag{4}$$

where

$$\Delta P_i = P_i^{sp} - E_i \sum_{j=1}^{n} (G_{ij}E_j - B_{ij}F_j) - F_i \sum_{j=1}^{n} (G_{ij}F_j + B_{ij}E_j) \qquad i \in n_{PQ} + n_{PV}$$

$$\Delta Q_i = Q_i^{sp} - F_i \sum_{j=1}^{n} (G_{ij}E_j - B_{ij}F_j) + E_i \sum_{j=1}^{n} (G_{ij}F_j + B_{ij}E_j) \qquad i \in n_{PQ}$$

$$\Delta V_i = |\bar{V}_i^{sp}| - (E_i^2 + F_i^2)^{1/2} \qquad i \in n_{PV}$$

4. APPLICATION OF THE SGA TO THE LOAD FLOW PROBLEM

Before applying the SGA to the load flow problem, some factors about the implementation of the GA have to be considered:

(1) coding the variables into a finite string;

(2) treatment of constraints;

(3) mapping the objective function to a fitness form.

The variables of the load flow problem are coded in the following manner. Firstly, each variable X is coded as a substring of length l which is interpreted as the usual unsigned binary integer on the interval $[0, 2^l - 1]$. Next this integer is mapped linearly to the variable's specified interval of the real numbers, $[X_{min}, X_{max}]$. Then, the substrings are concatenated to construct a multivariable string. The total string length L is thus $2(n-1)l$.

The constraints of the load flow problem are used as the specific intervals $[X_{min}, X_{max}]$ required in coding the variables. Thus the constraints will never be violated.

The load flow objective function to be minimized, $g(E, F)$, is transformed and normalized to a fitness scheme to be maximized:

$$f(E, F) = \frac{1}{1 + g(E, F)} \tag{5}$$

The system model, string coding, fitness function and simple genetic algorithm comprising a variant of a selection called the *stochastic remainder* selection [11], simple crossover and simple mutation have been programmed in Pascal. The program was operated on a PC computer with Intel 80386, 20 MHz. The algorithm was applied to the Klos–Kerner three-bus system [21] and the Ward–Hale six-bus system [22].

4.1. Klos–Kerner three-bus system

The Klos–Kerner three-bus system has four different solutions, one of which is the normal solution and the three others are abnormal solutions.

The variables of the Klos–Kerner three-bus system are E_2, F_2 and E_3, F_3. Each of these variables was coded as a 10-bit length substring in the interval $[-1.5, 1.5]$. The total string length was thus 40 bits. The following genetic parameters were used for the runs:

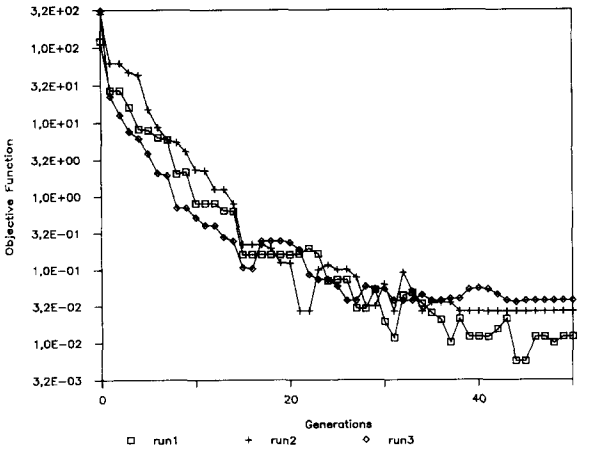

Fig. 1. Evolution of the objective function for finding an abnormal solution in the Klos–Kerner three-bus system.

population size = 100
probability of crossover = 0.9
probability of mutation = 0.01

The computation time for each run was about 1 min 40 s. Figure 1 shows the evolution of the objective function of the best individual in the population for three independent runs with different initial populations. It is seen in the Figure that the objective function decreases quickly from about 10^2 to some 10^{-2} in 30 generations. In fact, the near-optimal performance needs only about 3000 (30 generations × 100 strings) function evaluations in the search space where there are $2^{40} \cong 1.01 \times 10^{12}$ different configurations. This means that the SGA quickly finds a solution when exploiting a small portion of a very large search space. The best results of the three runs are listed in Table 1.

By comparing the above results with those in ref. 8, the results of runs 1 and 2 can be considered as being close to solution 4 and that of run 3 close to solution 3. The differences in the absolute values reside in the third decimal. At this stage, the SGA cannot do better owing to the bit representation of the variables. The discretization error is indeed

$$\Pi = \frac{X_{\max} - X_{\min}}{2^l - 1} = \frac{1.5 - (-1.5)}{2^{10} - 1} \cong 0.003$$

It is not surprising that the results obtained differ from the reference results at the third decimal.

As shown in Table 1, different solutions have been found by the runs with different initial populations. However, it was observed in the experiments that the chance of obtaining solution 4 is much greater than that of the other solutions and the normal solution 1 has never been found so far. In fact, the load flow function $g(E, F)$ is a kind of 'trap' function whose shape in the region of the normal solution is 'sharp'. By contrast, the regions of abnormal solutions are relatively 'flat' (see Fig. 3). Thus the GA is more likely to be trapped in the basin of the abnormal solutions and leads away from the normal one. In order to obtain the normal solution, some additional constraints have to be incorporated.

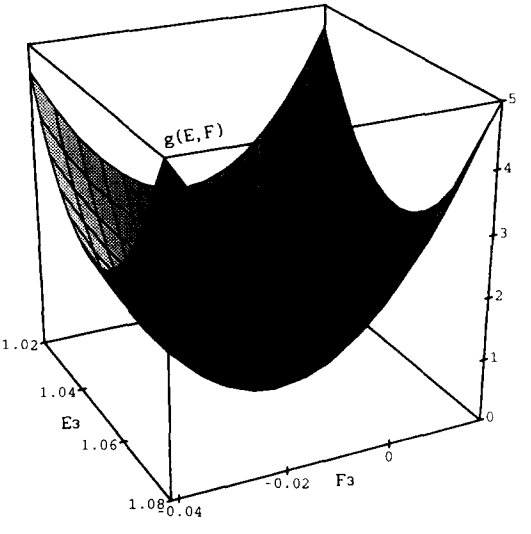

Fig. 2. Profile of the first solution (with E_2, F_2 fixed at the solution values).

TABLE 1

Best results of the abnormal solutions in the Klos–Kerner three-bus system

	E_2	E_3	F_2	F_3	$f(E, F)$	$g(E, F)$
Run 1	0.0073	0.0015	−0.0103	−0.0044	0.9943	0.0057
Run 2	0.0073	−0.0015	−0.0103	−0.0044	0.9738	0.0270
Run 3	0.5029	0.0015	0.0015	−0.0044	0.9635	0.0379

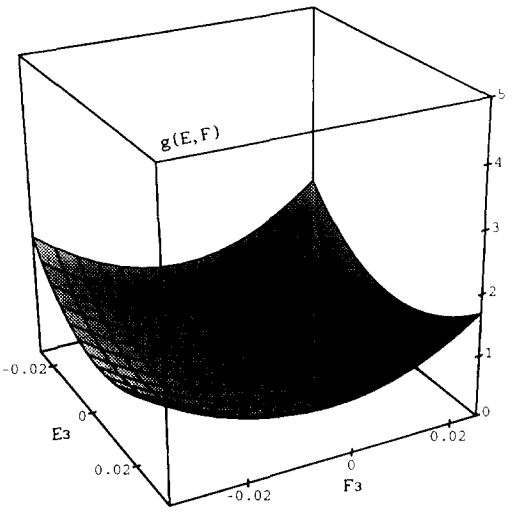

Fig. 3. Profile of the third solution (with E_2, F_2 fixed at the solution values).

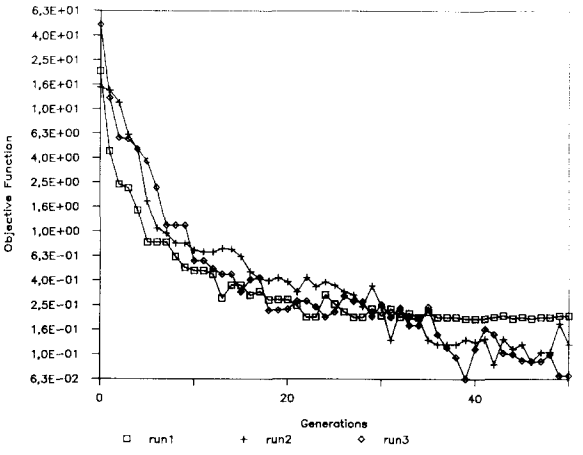

Fig. 4. Evolution of the objective function for finding an abnormal solution in the Ward–Hale six-bus system.

4.2. *Ward–Hale six-bus system*

Two solutions have been found by Tamura *et al.* [8] in the Ward–Hale six-busbar system in the case where the tap ratios are not taken into account, solution 1 being the normal solution and solution 2 an abnormal one.

In the Ward–Hale six-bus system, there are ten variables to solve: E_2, E_3, E_4, E_5, E_6 and F_2, F_3, F_4, F_5, F_6. On the PQ nodes, the variables were specified in the intervals $[-1.5, 1.5]$ for E and $[-1.5, 0]$ for F. Since the variations of E and F on PV nodes cannot exceed their specified magnitudes, E_2 and F_2 on the PV node were constrained in the intervals $[-1.1, 1.1]$ and $[-1.1, 0]$. The variables E and F were coded with 5 bits and 4 bits, respectively; the total string length was thus 45 bits. The following genetic parameters were used:

population size = 200
probability of crossover = 0.9
probability of mutation = 0.01

Each run required about 6 min 30 s for 50 generations. The performances of the algorithm are shown in Fig. 4. The best results of the three runs are listed in Table 2.

The results obtained seem to be rather crude owing to the discretization errors near 0.1 for E and F. However, by comparing them with the results of ref. 8, those obtained can be considered as the approximated solution 2. By using them as the initial voltage values of the Newton–Raphson method with optimal multiplier, all three runs converged to the exact solution 2 which was found by Tamura *et al.* To recall their method, the initial starting values were produced by a special systematic procedure and used repeatedly in a Newton–Raphson calculation. In the Ward–Hale six-bus case, 31 initial guesses were so generated, among which only the initial values no. 16 and no. 24 converged to solution 2. However, it appears that these initial values are far from the solution value. It seems that there does not exist an obvious relation between the initial values and the results obtained. By contrast, the results obtained by the SGA are already located in the promising solution region. Thus, using them as initial values, the Newton–Raphson method converges with certainty to the exact solution values. The results obtained by the SGA can serve as good guides for solution searches.

On the other hand, by running the SGA many times, it was always the abnormal solution 2 that was found. The conclusion concerning the $g(E, F)$ around the multiple solutions drawn from the Klos–Kerner three-bus system can be extended to the Ward–Hale six-bus system. This explains why, in our tests, the SGA falls vulnerably into the region of abnormal solution 2.

5. NORMAL LOAD FLOW SOLUTION SEARCH

It is known that the normal solution has voltage magnitudes near nominal voltage values and its voltage angles are smaller than

TABLE 2

Best results of the abnormal solutions in the Ward–Hale six-bus system

Bus no.	Run 1		Run 2		Run 3	
	E	F	E	F	E	F
2	0.2484	−1.1000	0.3194	−0.8800	−0.6032	−1.0267
3	0.3387	−0.5000	0.3387	−0.3000	0.0484	−0.4000
4	0.4355	−0.4000	0.4355	−0.2000	0.2419	−0.3000
5	0.1452	−0.6000	0.0484	−0.3000	−0.1452	−0.5000
6	0.2419	−0.4000	0.0484	−0.2000	0.0484	−0.3000
$f(E, F)$	0.8367		0.9236		0.9404	
$g(E, F)$	0.1952		0.0827		0.0634	

static stability limits (generally 90°). A system running under these conditions has very small losses. By contrast, the other solutions are characterized by some extremely low voltage magnitudes and/or voltage angles well beyond the static stability limits. These solutions represent abnormal system configurations (short-circuits, circulation of power around a transmission loop, heavy load states, loss of power or voltage stability, etc.). In consequence, their corresponding power losses are much greater than those of the normal solution.

In order to favor the normal solution, we have thus incorporated a power loss penalty term into the fitness function. In this way, the GA maximizes the fitness function not only by forcing the objective function (power mismatches) to zero but also constraining the power loss to a minimum value. The fitness function thus becomes

$$f(E, F) = \frac{1}{1 + g(E, F) + WP_{\text{loss}}{}^2} \tag{6}$$

where W is a penalty factor that scales the contribution of each term. The real power loss P_{loss} is determined by summing all the real power injections of the network buses:

$$P_{\text{loss}} = \sum_{i=1}^{n} \sum_{j=1}^{n} G_{ij}(E_i E_j + F_i F_j) \tag{7}$$

The genetic algorithm with this fitness scheme has been applied to the Klos–Kerner three-bus system and the Ward–Hale six-bus system. The same variable specifications and genetic parameters as previously used for these two systems were applied. The variations of the objective function $g(E, F)$ and of the real power losses in the best individual are shown in Figs. 5 and 6 for the Klos–Kerner three-bus system. The objective func-

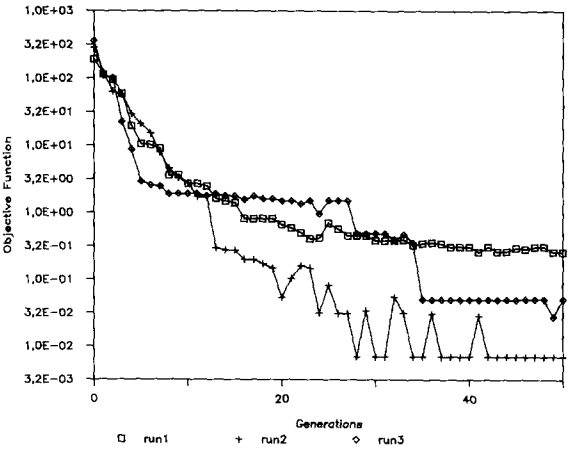

Fig. 5. Evolution of the objective function for finding the normal solution in the Klos–Kerner three-bus system.

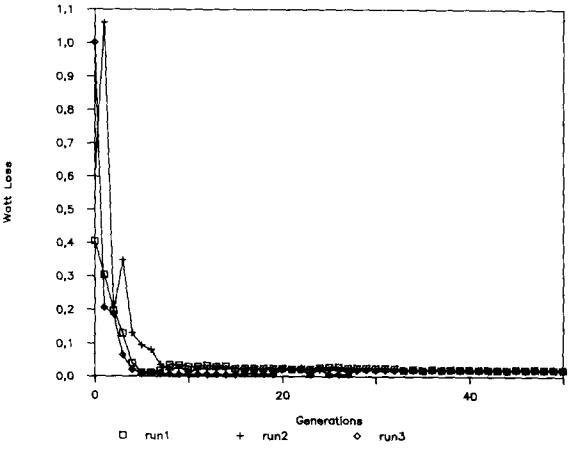

Fig. 6. Evolution of the real power loss: Klos–Kerner three-bus system.

tion decreases rapidly from the order of 100 to 1 and the corresponding real power losses fall to nearly zero in the first ten generations. The best string values of these runs are shown in Table 3.

The results were compared with solution 1 of the results of ref. 8; the largest absolute differences reside on the second decimal. Thus, the SGA finds the approximate normal solution starting from a random population rather than from the flat state used in traditional methods.

Similar results for the Ward–Hale six-bus system are shown in Figs. 7 and 8. The best results of each run are listed in Table 4. Although the values of the objective functions are not very close to zero, the solution values are shown to be in the promising region of the first solution. The results obtained may be improved by running some iterations of the Newton–Raphson method, starting with the above results.

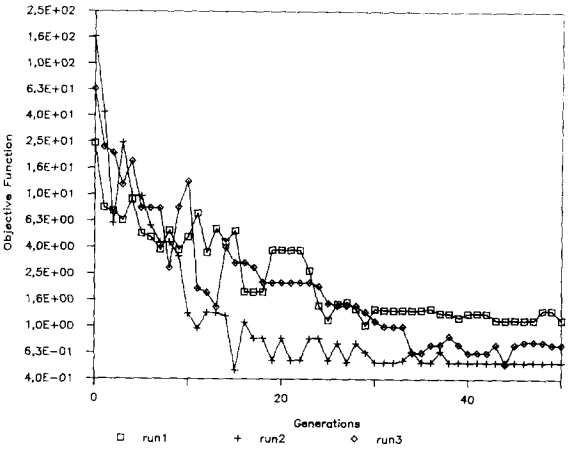

Fig. 7. Evolution of the objective function for finding the normal solution in the Ward–Hale six-bus system.

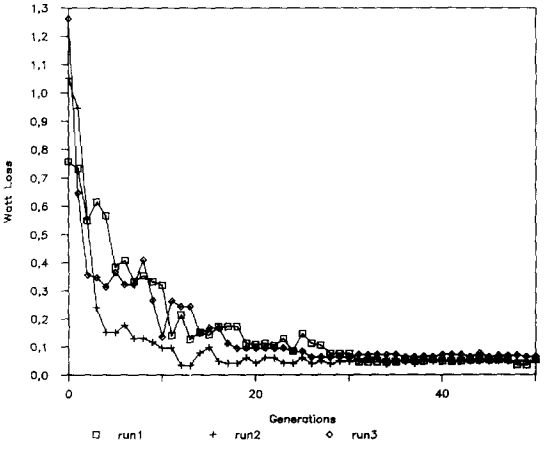

Fig. 8. Evolution of the real power loss: Ward–Hale six-bus system.

6. SIMULTANEOUS SEARCH FOR MULTIPLE LOAD FLOW SOLUTIONS

In the previous sections, the SGA was applied to search for the load flow solution and proved effective. From another point of view, it would be desirable that the population converges to a situation giving a proportional quota of individuals to all possible peaks of the objective function. The fact that all the individuals converged prematurely to a single peak is caused by the so-called 'genetic drift' [11] which is associated with early loss of diversity of the finite population. Function evaluations based on finite samples inevitably result in a stochastic error with respect to the desired situation. Repeated iterations of the algorithm accumulate the stochastic error, which leads the population to converge to one solution or another.

Since genetic algorithms are essentially derived from natural genetics, some natural remedies may combat the above problem. In nature, when many different species exist, competition between them does not take place in a 'face to face' manner. Instead, nature forms stable subpopulations of organisms surrounding separate niches by forcing similar individuals to share the available resources.

TABLE 3

Best results of the normal solution in the Klos–Kerner three-bus system

	E_2	E_3	F_2	F_3	$g(E, F)$	P_{loss}
Run 1	1.0836	1.0513	0.0015	−0.0015	0.2583	0.0205
Run 2	1.0806	1.0513	−0.0161	−0.0103	0.0070	0.0202
Run 3	1.0836	1.0543	−0.0161	−0.0103	0.0275	0.0218

TABLE 4

Best results of the normal solution in the Ward–Hale six-bus system

Bus no.	Run 1		Run 2		Run 3	
	E	F	E	F	E	F
2	0.8161	−0.2933	1.0290	−0.0	0.8161	−0.0733
3	0.8226	−0.2000	0.9194	−0.1000	0.7258	−0.2000
4	0.8226	−0.1000	0.9194	−0.1000	0.8226	−0.1000
5	0.7258	−0.2000	0.9194	−0.1000	0.7258	−0.1000
6	0.8226	−0.1000	0.9194	−0.1000	0.8226	−0.1000
$g(E, F)$	1.0216		0.5356		0.5218	
P_{loss}	0.0749		0.0494		0.0771	

Several methods have been implemented to induce this niche formation in genetic algorithms. In ref. 23, a *crowding* method [24] and a *sharing function* method [25] were presented and compared. It was concluded that the sharing function method was better able to allocate individuals to the peaks than the crowding method.

In the sharing concept, the population is divided into different subpopulations according to the similarity of the individuals in the population. A sharing function is defined to determine the neighborhood and degree of sharing for each string pair in the population. The neighborhood is measured by the distance between two individuals of the population, for example, a Euclidian distance in the decoded parameter space (x variables) or a Hamming distance in the gene space (strings).

A power-law sharing function $\text{Sh}(d_{ij})$ as a function of the distance d_{ij} between two individuals i and j is expressed as

$$\text{Sh}(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{\text{sh}}}\right)^{\alpha} & \text{if} \quad d_{ij} < \sigma_{\text{sh}} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where σ_{sh} and α are constant. The parameter σ_{sh} is defined as the maximum distance between strings which is necessary to form as many niches as there are peaks in the solution space.

From formuls (8), the strings close to an individual have a high degree of sharing (close to 1) and the strings far from the individual have a very small degree of sharing (close to 0). Thus the degree of sharing for individual i is determined by summing the sharing function values contributed by all N strings in the entire population. This gives the approximate number of individuals in the same niche, m_i':

$$m_i' = \sum_{j=1}^{N} \text{Sh}(d_{ij}) \tag{9}$$

To implement the idea of sharing, an individual's shared fitness f_i' is defined as its original fitness f_i divided by its niche count m_i':

$$f_i' = f_i / m_i' \tag{10}$$

In this way, when many individuals are in the same neighborhood, they contribute to a high value of the niche count m_i', thereby derating their fitness. As a result, the different species are obliged to share the limit resources.

There are two types of sharing due to the two ways of defining the distance. The sharing based on the proximity of the individuals in the decoded parameter space is called *phenotypic* sharing, while the sharing based on the proximity of the individuals in the gene space is called *genotypic* sharing. These two types of sharing were compared in ref. 23; the performance of phenotypic sharing was shown to be better than genotypic sharing.

The genetic algorithm with the phenotypic sharing scheme has been tested on the Klos–Kerner three-bus system. As before, the simple genetic algorithm with stochastic remainder selection, simple crossover and simple mutation was implemented in a program. In the sharing scheme, the triangular sharing distribution ($\alpha = 1$) was used and σ_{sh} was taken as 0.25. The distance between individuals was calculated using the Euclidian distance. Each variable was represented by a 9-bit substring

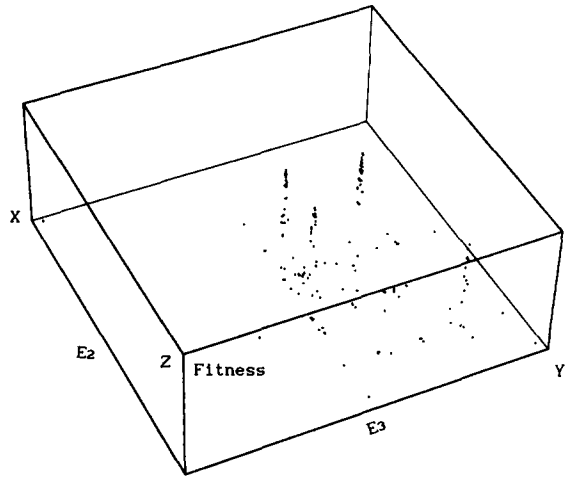

Fig. 9. Point distribution in the 50th generation.

in the interval $[-1.5, 1.5]$. The total string length was then 36 bits. The following genetic parameters were used:

population size = 200
probability of crossover = 0.9
probability of mutation = 0.01

It took about 23 min for 50 generations. Figure 9 shows the distribution of 200 individuals at the 50th generation in three-dimensional space where X and Y represent E_2 and E_3, respectively, in the interval $[-1.5, 1,5]$; the Z axis represents the fitness height in the range $[0, 1]$.

Four stable clusters of points can be seen in Fig. 9. The best individual values, the shared fitness, the original fitness, and the approximate number of individuals in each subpopulation are listed in Table 5. We can easily verify that the best string values of Table 5 are in fact the approximate four solutions of the Klos–Kerner three-bus system. The number of individuals in each niche is quasi-proportional to its original fitness height. As a result, the shared fitnesses of the four clusters differ little from each other.

TABLE 5

The best individuals of four subpopulations

	Clust. 1	Clust. 2	Clust. 3	Clust. 4
E_2	1.0949	−0.0029	0.5078	−0.0029
E_3	1.0597	0.6781	0.0029	0.0029
F_2	−0.0029	−0.0029	−0.0088	−0.0088
F_3	0.0029	0.0029	0.0029	0.0029
Shared fitness	0.0352	0.0266	0.0285	0.0256
Original fitness	0.6972	0.8687	0.9347	0.9198
Objective function	0.4342	0.1511	0.0699	0.0872
Niche count	20	32	33	36

Intuitively, there should be an equal number of individuals located on each of the four peaks. However, the niche counts of solutions 2, 3, and 4 are of the same order, while the niche count of solution 1 is smaller than the others. In addition, the highest fitness value of solution 1 is smaller than the peak values of the other solutions. This is due firstly to the finite sampling. Stochastic errors always exist, which inevitably results in some undesirable errors. The other reason is associated with the various $g(E, F)$ profiles of the different solutions. As discussed previously, although each solution peak has equal magnitude, their profiles are not the same. The region round the normal solution is rather sharp and the regions round the abnormal solutions are relatively flat. As a result, it should be easier to locate individuals in the abnormal solutions than to locate them in the normal one.

The sharing function has also been tested on the Ward–Hale six-bus system. Successive generations show that the effect of sharing is not very significant. Only the approximated solution 2 was found at the end of the process. The niche surrounding the normal solution was not even formed, probably due to the profile of the load flow function, the crude variable representation, or an insufficient population size. Further investigations should be carried out to circumvent this problem.

7. IMPROVING THE PRECISION OF THE GA RESULTS

It is appreciated from the previous sections that the GA efficiently finds the approximate solutions or the region of the solutions. Nevertheless, the results obtained are very crude compared with the results obtained by some traditional methods. One possible remedy is to switch to the traditional methods when the individuals of the population are close to the exact solution, as we have done with the Newton–Raphson method. However, this traditional method inevitably requires computa-

tion of the Jacobian matrix and this is in contradiction with what has been promised for GAs. In this section, a new technique which does not require calculation of derivatives is introduced and demonstrated by solving the normal solution of the Klos–Kerner three-bus system.

It was shown in the previous experiment that the population evolved rapidly in the first generations; after that, it became quasi-stable and future improvement of fitness seems to be difficult. This is firstly because the individuals of the population become too similar. Better individuals are not well distinguished from the others. Secondly, the discretization errors related to the strings prevent the GA exploring precisely the solution space. To encourage competition between individuals, the difference in their fitness may be amplified artificially. The following fitness scheme is then adopted:

$$f(E, F) = \frac{1}{1 + Kg(E, F) + WP_{\text{loss}}{}^2} \tag{11}$$

where K is an adaptation factor used to expand the objective function scale when approaching its minimal value. The fitness scheme (11) is in fact a generalization of the previous ones. For $K = 1$, the fitness scheme (6) is obtained; for $K = 1$ and $W = 0$, the fitness scheme (5) is obtained.

Simultaneously, for a fine tuning of the strings, the constraints on the variables may be tightened. This process is repeated until the end of the evolution process.

As a demonstration of the above ideas, the search for the normal solution in the Klos–Kerner three-bus system was taken again. The initial values of K and W were, respectively, 1 and 50. Each time the maximum fitness became greater than 0.9 in the successive generations, the factor K was multiplied by 10. The penalty factor W was then reduced to zero, since at that moment the individuals had been concentrated in the region of the normal solution. For each of three successive occurrences of maximum fitness 0.9, the bounds on the variables were successively reduced to $\pm\beta = \pm 0.1$, then to $\pm\beta/2$, and finally to $\pm\beta/4$ around the current best string values. A reinitialization of the population was applied only once at the first occurrence of fitness 0.9. The program ran with the same genetic parameters and the same initial populations as previously. The performance of this technique is shown in Fig. 10. The diagram shows that the technique did not play a role in the first run because the maximum fitness of the population never reached 0.9 (this also means that the objective function decreases to under 0.1), which is the criterion for starting the technique. In runs 2 and 3, the objective function had decreased to values of the order of 10^{-5} to 10^{-7} after 50 generations. The computation time remained almost the same as previously. The best results of the three runs are listed in Table 6.

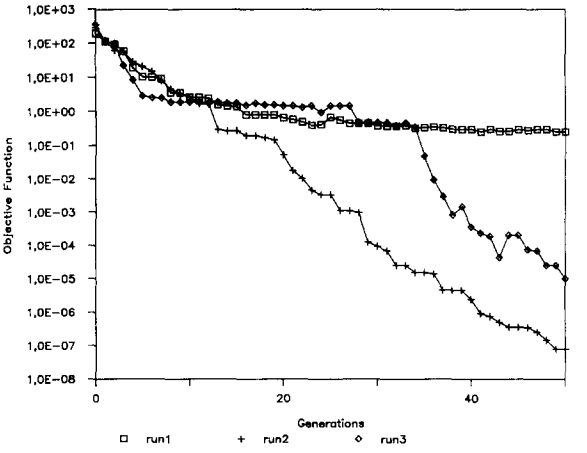

Fig. 10. Evolution of the objective function with use of the technique to improve the GA's precision.

The results of runs 2 and 3 are very precise compared with the reference results. However, run 1 implies that this technique is not uni-

TABLE 6

The best three run results using the technique for improving the GA results

	E_2	E_3	F_2	F_3	$g(E, F)$
Run 1	1.0836	1.0513	0.0015	−0.0015	0.2583
Run 2	1.0780	1.0499	−0.0184	−0.0114	0.8×10^{-7}
Run 3	1.0780	1.0498	−0.0183	−0.0114	1.0×10^{-5}

formly efficient for all cases. The method seems promising, but ought to be somewhat more robust.

8. CONCLUSION

In this introductory paper, the application of genetic algorithms to the load flow problem has been examined.

Without any requirements for auxiliary information and calculation of derivatives, a simple algorithm based on the reproduction, crossover, and mutation operations quickly finds an approximate abnormal solution after exploring quite a small portion of the large search space.

After discussing some physical characteristics of multiple load flow solutions and by incorporating a penalty term based on the system losses into the GA's fitness scheme, the normal load flow solution has been found easily without changing the fundamental principle of the algorithm.

An attempt to search all solutions simultaneously by the SGA with a sharing function has been made. Test results on the Klos–Kerner three-bus network are promising. At present, the computation is time-consuming owing to the fact that calculation of the niche count is of complexity $N \times N$. One of our future projects will use the so-called 'cluster analysis methods' [26] for which the complexity is expected to be $k \times N$, where k is the number of solutions. Moreover, the σ_{sh} parameter needs to know *a priori* the number of solutions, which is a difficult problem. The theoretical results of ref. 7 are based on too particular assumptions and do not seem precise enough for the purpose. By contrast, cluster analysis algorithms could be helpful in revealing this number.

An adaptive technique which does not require derivative information has also been presented for improving the accuracy of the results.

Since genetic algorithms are highly parallelizable, it is believed that they will be applied increasingly with parallel computers.

ACKNOWLEDGEMENTS

The authors wish to thank Professors G. Labbé and N. Janssens for many fruitful discussions and support of various aspects of the research.

REFERENCES

1 B. Stott, Review of load-flow calculation methods, *Proc. IEEE, 62* (1974) 916 - 929.
2 S. Abe, N. Hamada, A. Isono and K. Okuda, Load flow convergence in the vicinity of a voltage stability limit, *IEEE Trans., PAS-97* (1978) 1983 - 1993.
3 A. J. Korsak, On the question of the uniqueness of stable load flow solutions, *IEEE Trans., PAS-91* (1972) 1093 - 1100.
4 A. Klos, S. Mikolajczyk and A. Kaminaki, Non-uniqueness and stability of load flow, *Proc. 6th Power System Computation Conf. (PSCC), Darmstadt, F.R.G., 1978*, IPC, Guildford, U.K., pp. 704 - 710.
5 B. K. Johnson, Extraneous and false load flow solutions, *IEEE Trans., PAS-96* (1977) 524 - 534.
6 Y. Tamura, H. Mori and S. Iwamoto, Relationship between voltage instability and multiple load flow solutions in electrical power systems, *IEEE Trans., PAS-102* (1983) 1115 - 1125.
7 J. Baillieul and C. I. Byrnes, Geometric critical point analysis of lossless power system models, *IEEE Trans., CAS-29* (1982) 724 - 737.
8 Y. Tamura, K. Iba and S. Iwamoto, A method for finding multiple load-flow solutions for general power systems, *IEEE PES Winter Meeting, New York, 1980*, Paper No. A 80-043-0.
9 K. Iba, H. Suzuki, M. Egawa and T. Watanabe, A method for finding a pair of multiple load flow solutions in bulk power systems, *IEEE Trans., PWRS-5* (1990) 582 - 591.
10 J. H. Holland, *Adaptation in Natural and Artificial Systms*, Univ. Michigan Press, Ann Arbor, MI, 1975.
11 D. E. Goldberg, *Genetic Algorithms in Searching, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
12 D. E. Goldberg and C. H. Kuo, Genetic algorithms in pipeline optimization, *J. Comput. Civil Eng., 1* (1987) 128 - 141.
13 L. Davis and D. Smith, Adaptive design for layout synthesis, *Internal Rep.*, Texas Instruments, Dallas, TX, 1985.
14 L. Davis and S. Coombs, Optimizing network link sizes with genetic algorithms, in M. S. Elzas, T. I. Ören and B. P. Zeigler (eds.), *Modelling and Simulation Methodology: Knowledge Systems' Paradigms*, North-Holland, Amsterdam, 1989, pp. 317 - 331.
15 J. K. Parker, A. R. Khoogar and D. E. Goldberg, Inverse kinematics of redundant robots using genetic algorithms, *Proc. Int. Conf. Robotics and Automation, 1989*, IEEE, New York, pp. 271 - 276.
16 R. Das and D. E. Goldberg, Discrete-time parameter estimation with genetic algorithm, in W. G. Vogt and M. H. Mickle (eds.), *Proc. 19th Annu. Pittsburgh Conf. Modeling and Simulation, Part 5: Control, Robotics, Univ. Pittsburgh, PA, 1988*, pp. 2391 - 2395.
17 D. E. Glover, Solving a complex keyboard configuration problem through generalized adaptive search, in L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987, pp. 12 - 31.

18 J. M. Fitzpatrick, J. J. Grefenstette and D. Van Gucht, Image registration by genetic search, *Proc. IEEE Southeast CON 84, Louisville, KY, 1984*, pp. 460 - 464.

19 D. E. Goldberg and M. P. Samatani, Engineering optimization via genetic algorithm, in K. M. Will (ed.), *Proc. 9th Conf. Electronic Computation, Univ. Alabama at Birmingham, AL, 1986*, Am. Soc. Civil Eng., 1986, pp. 471 - 482.

20 L. Davis and M. Steenstrup, Genetic algorithms and simulated annealing: an overview, in L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987, pp. 1 - 11.

21 A. Klos and A. Kerner, The non-uniqueness of load-flow solutions, *Proc. 5th Power Systems Computation Conf. (PSCC), Cambridge, U.K. 1975*, p. 3.1/8.

22 J. B. Ward and H. W. Hale, Digital computer solution of power-flow problems, *Trans. AIEE, PAS-75* (1956) 398 - 404.

23 K. Deb and D. E. Goldberg, An investigation of niche and species formation in genetic function optimization, in J. D. Schaffer (ed.), *Proc. 3rd Int. Conf. Genetic Algorithms, 1989*, Kaufmann, 1989, pp. 42 - 50.

24 K. A. Dejong, An analysis of the behavior of a class of genetic adaptive systems, *Doctoral Dissertation*, Univ. Michigan, Ann Arbor, MI, 1975.

25 D. E. Goldberg and J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in J. J. Grefenstette (ed.), *Proc. 2nd Int. Conf. Genetic Algorithms, 1987*, Erlbaum, 1987, pp. 41 - 49.

26 M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973.