Genetic Algorithms for Control and Signal Processing

K.F. Man and K.S. Tang
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong
email: k.man@ieee.org

Abstract—**The practical application of Genetic Algorithms (GA) to the solution of engineering problem is a rapidly emerging approach in the field of control engineering and signal processing. This tutorial is to provide a comprehensive coverage of the techniques involved, describing the characteristics, advantages and constraints of GA, as well as discussing genetic operations such as crossover, mutation and reinsertion. The intrinsic characteristics in term parallelism, multiobjective, and multimodal etc. will be outlined. The features of this approach are illustrated by real-world applications. Also described is a newly proposed and unique hierarchical genetic algorithm designed to address the problem in determining system topology.**

I. BASIC CONCEPTS OF GA

The basic principles of GA were first proposed by Holland [23]. Thereafter, a series of literatures [9], [17], [33], [37] and reports [32], [54] have become available. GA is inspired by the mechanism of natural selection where stronger individuals would likely be the winners in a competing environment. Here, GA uses a direct analogy of such natural evolution. It presumes that a potential solution of a problem is an individual and can be represented by a set of parameters. These parameters regarded as the genes of a chromosome and can be structured by a string of values in binary form. A positive value, generally known as fitness value, is used to reflect the degree of "goodness" of the chromosome for the problem which would be highly related with its objective value.

Through out a genetic evolution, the fitter chromosome has the tendency to yield good quality offspring which means a better solution to the problem. In a practical application of GA, a population pool of chromosomes has to be installed and they can be randomly set initially. The size of this population varies from one problem to the other although some guidelines are given in [30]. In each cycle of genetic operation termed as evolving process, a subsequent generation is created from the chromosomes in the current population. This can only be succeeded if a group of those chromosomes, generally called "parents" or a collection term "mating pool" are selected via a specific selection routine. The genes of the parents are to be mixed and recombined for the production of offspring in the next generation. It is expected that from this process of evolution (manipulation of genes), the "better" chromosome will create a larger number of offspring, and thus has a higher chance of surviving in the subsequent generation, emulating the survival-of-the-fittest mechanism in nature.

A scheme called Roulette Wheel Selection [9] is one of the commonly techniques being used for such an proportionate selection mechanism. To illustrate this further, the selection procedure is listed in Table I.

TABLE I
ROULETTE WHEEL PARENT SELECTION

- Sum the fitness of all the population members; named as total fitness (N).
- Generate a random number (n) between *0* and N
- Return the first population member whose fitness added to the fitness of the preceding population members, is greater than or equal to n.

The cycle of evolution is repeated until a desired termination criterion is reached. This criterion can also be set by the number of evolution cycles (computational runs), or the amount of variation of individuals between different generations, or a pre-defined value of fitness.

In order to facilitate the GA evolution cycle, two fundamental operators: Crossover and Mutation are required for such a process, although the selection routine can be termed as the other operator. To further illustrate the operational procedure, an one-point Crossover mechanism is depicted on Fig. 1. A crossover point is randomly set. The portions of the two chromosomes beyond this cut-off point to the right is to be exchanged to form the offspring. An operation rate (p_c) with typical value between 0.6 and 1.0 is normally used as the probability of crossover.

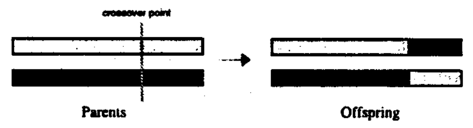


Fig. 1. Example of One-Point Crossover

Whereas for mutation, it is applied to each offspring individually after crossover exercise. It alters each bit randomly with a small probability (p_m) with a typically value less than 0.1.

The choice of p_m, p_c as the control parameters can be a complex nonlinear optimization problem to be solved.

0-7803-3932-0

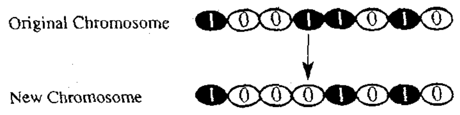


Fig. 2. Bit Mutation on the fourth bit

Furthermore, their settings are critically dependent upon the nature of the objective function. This selection issue still remains open to suggestion although some guidelines have been introduced by [20]:

- For large population size (100)
  $p_c = 0.6$ and $p_m = 0.001$
- For small population size (30)
  $p_c = 0.9$ and $p_m = 0.01$

Fig. 3 summarizes the standard genetic algorithm.

```
Standard Genetic Algorithm ()
{
  // start with an initial time
  t := 0;
  // initialize a usually random population of individuals
  initpopulation P (t);
  // evaluate fitness of all initial individuals of population
  evaluate P (t);
  // test for termination criterion (time, fitness, etc.)
  while not done do
        // increase the time counter
        t := t + 1;
        // select a sub-population for offspring production
        P' := selectparents P (t);
        // recombine the "genes" of selected parents
        recombine P' (t);
        // perturb the mated population stochastically
        mutate P' (t);
        // evaluate it's new fitness
        evaluate P' (t);
        // select the survivors from actual fitness
        P := survive P,P' (t);
  od
}
```

Fig. 3. Standard Genetic Algorithm

II. MODIFICATION OF GA

The GA mechanism is neither governed by the use of the differential equations nor does it rely on any continuous function. However, it possesses the unique ability to search and optimize a solution for a complex system, where other mathematical oriented techniques may have failed to compile the necessary design specifications. Due to its evolutionary characteristics, a standard GA may not be flexible enough for a practical application, and an engineering insight is always required whenever a GA is applied. As the problem becomes complicate, multi-tasking and conflicting, the structure of GA has to be altered to suit the requirement. There are many facets of operational modes that can be introduced on the chromosomes, the operators and the implementation.

A. Chromosome Representations

Bit string encoding [23] is the most classical approach used by GA researchers for its simplicity and traceability. A minor modification is the use of Gray code in the binary coding. [24] investigated the use of GA for optimizing functions of two variables and claimed that a Gray code representation works slightly better than the normal binary representation.

Recently, the direct manipulation of real-value chromosomes [26], [57] raised some considerable of interest. This representation was introduced especially to deal with real parameter problems. The work currently being taken place by [26] indicates that the floating point representation would be faster in computation and more consistent from the basis in run-to-run. However, the opinion given by [18] suggested that a real-coded GA would not necessarily yield good result in some situations despite many practical problems have been solved by using real-coded GA. So far, there is insufficient consensus to be drawn on this argument.

String-based representation may poise difficulty and sometimes unnatural to some optimization problems. The use of other encoding techniques, such as order-based representation [9] (for bin-patching, graph colouring), embedded lists [37] (for factory scheduling problems), variable element lists [37] (for semiconductor layout), and even LISP S-expressions [28] can thus be explored.

B. Objective and Fitness Value

The objective function of a problem is a main source to provide the mechanism for evaluating the status of each chromosome. This is an important link between GA and the system. It takes chromosome as input and produces a number or list of numbers (objective value, generally in least square form) as a measure to the chromosome's performance.

However, its range of values varies from problem to problem. To maintain uniformity over various problem domains, objective value is re-scaled to a fitness value by linear scaling, power law scaling, or sigma truncation [17].

C. Selection Mechanisms

To generate good offspring, a good parent selection mechanism is necessary. This is a process used for determining the number of trials for one particular individual used in reproduction. The chance of selecting one chromosome as a parent should be directly proportional to the number of offspring produced.

Stochastic Universal Sampling (SUS) is another single-phase sampling algorithm with time complexity of SUS is in the order of N [2], comparing with Roulette Wheel

Selection which in the order of $NlogN$. There are other methods can be used such as the Ranking scheme [1]. It introduces an alternative to proportional fitness assignment. The chromosomes are selected proportionally to their rank rather than actual evaluation values. It has been shown to help the avoidance of premature convergence and to speed up the search when the population approaches convergence [53].

D. Genetic Operations

D.1 Crossover

Although one-point crossover was inspired by biological processes, it has a major drawback that certain combination of schema cannot be combined in some situations [37]. A multi-point crossover can be introduced to overcome this problem. As a result, the performance of generating offspring is greatly improved. An example of this operation is depicted in Fig. 4 where multiple crossover points are randomly selected.

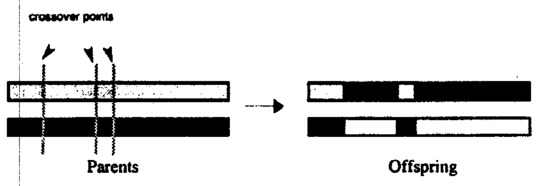


Fig. 4. Example of Multi-Point Crossover (m=3)

Another approach is the Uniform crossover. This generates offspring from the parents based on a randomly generated crossover mask. The operation is demonstrated in Fig. 5.

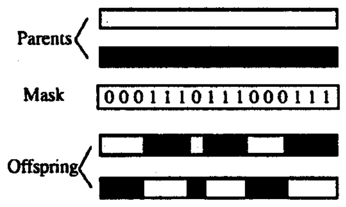


Fig. 5. Example of Uniform Crossover

The resultant offspring contains a mixture of genes from each parent. The number of effective crossing points is not fixed, but will be averaged at $L/2$ (where L is the chromosome length).

The preference of using which crossover techniques is still arguable [13], [42], [44]. A general comment is that each of these crossover is particularly useful for some classes of problems and quite poor for others, except that one-point crossover is indicated as a "loser" experimentally [13].

Crossover operations can be directly adopted into the chromosome with real number representation. The only difference would be the string is composed of a series of real numbers instead of binary number.

Some other problem based crossover techniques have been proposed. [17] described a Partially Matched Crossover (PMX) for the order-based problem. [6] designed an "analogous crossover" for the robotic trajectory generation. Therefore, the use of crossover technique to improve the offspring production, is very much problem oriented. All in all, there is no unified view on this front.

D.2 Mutation

Originally, mutation was designed only for the binary-represented chromosome. To adopt the concept of introducing variations into the chromosome, a random mutation [37] has been designed for the real number chromosome:

$$g = g + \psi(\mu, \sigma) \qquad (1)$$

where g is the real value gene; ψ is a random function which may be Gaussian or normally distributed; μ, σ are the mean and variance related with the random function, respectively.

E. Reinsertion

After generating the sub-population (offspring), there are several representative strategies to be proposed for old generation replacement.

In the case of generational replacement, the chromosomes in the current population is completely replaced by the offspring [20]. Therefore, the population with size N will generate N offspring in this strategy. One can argue that this strategy may make the best chromosome of the population fail to reproduce offspring in the next generation. So it is usually combined with elitist strategy where one or a few of the best chromosomes are copied into the succeeding generation. The elitist strategy may increase the speed of domination of a population by a super chromosome, but on balance it appears to improve the performance.

On the contract, when a small number of offspring are generated in the evolution cycle, a part of the current population is replaced by the new generated offspring. Usually, the worst chromosomes are replaced when new chromosomes are inserted into the population. However, a direct replacement of the parents by the corresponding offspring may also be adopted.

F. Probability Rates Setting

The choice of an optimal probability operation rates for crossover and mutation is another controversial debate for both analytical and empirical investigations. The increase of crossover probability would cause the recombination of building blocks to rise, and at the same time, it also increases the disruption of good chromosomes. On the other hand, should the mutation probability increases, this would transform the genetic search into a random search, but it helps to reintroduce the lost genetic material.

As each operator probability may vary through the generations, Davis [7] suggested a linear variations in crossover and mutation probability, with decreasing crossover rate during the run while mutation rate increases. Syswerda [45] imposed a fixed schedule for both cases but Booker [4] utilized a dynamically variable crossover rate which is depending upon the spread of fitness. [8], [9] modified the operator probabilities according to the success of generating "good" offspring.

G. Parallelism

One of the major criticism of using GA must be the time spent in computation. Sometime, it can be painfully long. This is understandable as GA is not a mathematical guided solution to the problem. It is merely a stochastic, discrete, nonlinear and highly dimensional search algorithm. To use GA for practical applications, particularly in control and signal processing areas, the problem of computing overrun time requires much attention to be resolved.

Considering that GA is already having an intrinsic parallelism architecture, in a nutshell, there requires no extra effort to construct a parallel computational framework. Rather, GA can be fully exploited in its parallel structure to gain the required speed for practical uses.

There are a number of GA-based parallel method to enhance the computational speed [5]. The methods of parallelization can be classified as Global, Migration [51] and Diffusion [3]. These categories reflect different ways in which parallelism can be exploited in GA as well as the nature of the population structure and recombination mechanisms used.

H. Multiple Objective

Without any doubt, the GA always has the distinct advantage of being able to solve multiobjective problems that other gradient type of optimizers have failed to meet. Indeed, engineering problems often exist in the class of multiple objectives. Historically, multiple objectives have been combined in an ad hoc manner so that a scalar objective function is formed for the usual linearly combined (weighted sum) functions of the multiple attributes [55]. Another way is by turning the objectives into constraints. There is a lot of work being done using the weighted sums and penalty functions to turn multiobjective problems into single-attribute problems even when a GA is applied.

There is another approach for searching the multi-attribute spaces [15] of these multiobjective problems. In this approach, the solution set of a multiobjective optimization problem consists of all those vectors such that their components cannot all be simultaneously improved. Various methods like Vector Evaluated GA (VEGA) [41], Pareto-based fitness assignment [14], [17], Pareto ranking with goal attainment [14], tournament selection based on Pareto dominance [25] have been reported.

It should be noticed that the use of Pareto-based evolutionary optimization may cause the finite populations converging to only one of these, due to stochastic errors in the selection process. To remedy this, the additional use of fitness sharing [14], [16] and mating restriction [14], [21] must be adopted.

III. GA APPLICATIONS

A. Active Noise Control (ANC)

A generic feedforward type of ANC system is shown in Fig. 6. It consists of four different parts: detector, error sensor, secondary source(s) and the controller. The detector and error sensor are electronic devices that are used for picking up the primary noise signal and error signal, respectively. The secondary sources, usually loudspeakers, are to generate the corresponding anti-phase signal for the noise signal, $s(k)$.

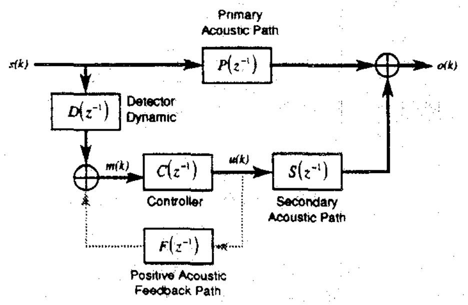


Fig. 6. Active noise control using feedforward control

The principle of noise cancellation is to ensure that the controller $C(z^{-1})$ produces the appropriate control signal $u(k)$, according to the reference signal $m(k)$, so that the resultant error signal $o(k)$ is minimized. In order to have a good cancellation effect, the estimates of the acoustic paths using the filter model should be accurately made in deriving the controller $C(z^{-1})$. An FIR filter, that has a unimodal error surface, is normally adopted for this purpose.

Considering that the ANC configuration is largely affected by the inherent nature of time delay, it is obvious that a large portion of the high order FIR filter is being used for the time delay modelling. Hence, a modified FIR filter that comprises a time delay element would be more appropriate than the usual high order FIR filter. As a result, a general form of the modified model for the estimation of the acoustic path process is proposed:

$$H(z^{-1}) = g \cdot z^{-d} \sum_{i=0}^{L-1} b_i z^{-n \cdot i} \qquad (2)$$

where g is the appropriate d.c gain, d is the time delay element, b_i is the filter coefficient, L is the number of taps, n is the tap separation.

It should be noted that the traditional LMS technique is ill suited to estimate this modified FIR model correctly, particularly for the variables of g and d, which are initially unknown and belong to the class of multimodal error surfaces for optimization.

To realize the system, two individual units known as the Real-Time Executed System (RTES) and the Genetic Algorithm Learning System (GALS) were developed. RTES is used to provide a speedy, but controllable, solution of the system while GALS optimizes and refines the controller, in order to achieve the required optimal noise control performance. Each system is implemented using a TMS320C30 processor together with its own local memory. The detailed description of RTES and GALS are given in [47]. The results are depicted in Fig. 7.

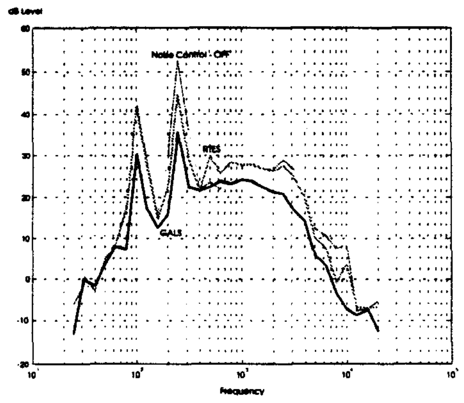


Fig. 7. Experimental results

B. Automatic Speech Recognition

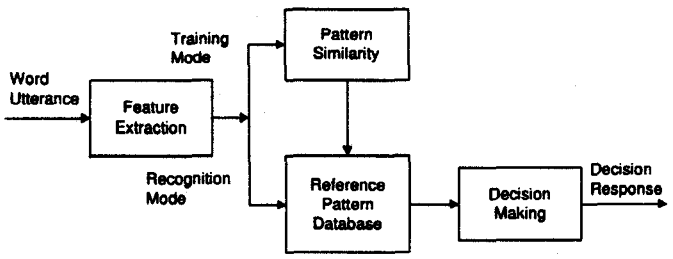


Fig. 8. Block Diagram of the Speech Recognition System

Another classic study to demonstrate the effectiveness of GA in engineering applications is the development of a Automatic Speech Recognition (ASR) system. Fig. 8 shows the overall ASR system that based on the template matching technique.

A typical ASR system usually runs in one of the two following modes:

1. the training mode; or
2. the recognition mode.

It is well known in speech recognition that individual speaking rates involve great variations [39]. These variations cause a nonlinear fluctuation, or timing difference of different utterances in the time-axis. Such a feature has been tackled by the use of a technique called dynamic time warping (DTW) [40]. This is a dynamic programming-matching (DP-matching) based nonlinear time-alignment scheme. DTW requires minimal system requirements, and is considered a faster training method but uses less searching times when compared with other available technique, such as the Hidden Markov Model. However, there are some drawbacks [38] which may degrade the performance of DTW. The obstacles are largely classified as follows:

- the stringent requirement of the slope weighting function;
- the non-trivial finding of the K-best paths; and
- the relaxed endpoint constraint.

To overcome these problems, a stochastic method called Genetic Time Warping (GTW) is developed. GTW is a method that searches the warping paths, instead of using DP-matching. In this way, the problems that are confronted with DTW can be eliminated.

B.1 Warping Path

In order to formulate the GTW scheme, the warping path model should first be defined. Let $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ represent two speech patterns, where x_i and y_i are the parametric vectors of the short-time acoustic features (any set of acoustic features can be used, as long as the distance measure for comparing pairs of feature vectors is known and its properties are well understood). We use i_x and i_y to denote the time indices of X and Y, respectively. The dissimilarity between X and Y is defined by a function of the short-time spectral distortions $d(x_{i_x}, y_{i_x})$, which will be denoted as $d(i_x, i_y)$ for simplicity, where $i_x = 1, 2, \ldots, N$ and $i_y = 1, 2, \ldots, M$.

A general time alignment and normalization scheme involves the use of two warping functions, ϕ_x and ϕ_y, which relate the indices of the two speech patterns, i_x and i_y, respectively. Both quantities are subjected to a common "normal" time axis k, i.e.,

$$\begin{aligned} i_x &= \phi_x(k) \\ i_y &= \phi_y(k) \quad \text{where} \quad k = 1, 2, \ldots, T \end{aligned}$$

A global pattern dissimilarity measures $d_\phi(X, Y)$ based on the warping path $\phi = (\phi_x, \phi_y)$ (warping function pair) as the accumulated distortion over the entire utterance can thus be defined as follows:

$$D_\phi(X, Y) = \sum_{k=1}^{T} d\left(\phi_x(k), \phi_y(k)\right) m(k) / M_\phi \tag{3}$$

where $d(\phi_x(k), \phi_y(k))$ is a short-time spectral distortion defined for the $x_{\phi_x}(k)$ and $y_{\phi_y}(k)$; $m(k)$ is a non-negative (path) weighting coefficient; and M_ϕ is a (path) normalizing factor.

This clearly shows that an extremely large number of possible warping function pairs can be found. To extract some possible paths from this function is only possible if some kind of distortion measurement is used. One natural and popular choice [40] is to adopt a dissimilarity function $D(X, Y)$ as the minimum of $D_\phi(X, Y)$ over a whole range of possible paths and this is achieved by minimizing Eqn. (3) such that

$$D(X, Y) = \min_\phi D_\phi(X, Y) \tag{4}$$

where the warping function (path) ϕ must satisfy a set of conditions [38].

By the same token, the warping path ϕ must also obey the properties of the actual time-axis. In other words, when the mapping of the time axis of a pattern X onto the time axis of a pattern Y is necessary, while the linguistic structures of the speech patterns X must also be preserved. These linguistic structures are characterized by the continuity, temporal order, local continuity and other acoustic information of the speech signals.

For a realistic warping path ϕ, the following criteria must be satisfied:

- the endpoint constraint;
- the local monotonic constraint;
- the local continuity;
- the slope constraint; and
- the allowable region.

B.2 Implementation of Genetic Time Warping

Having established the essential arguments for the formulations of the time warping problem, the actual implementation procedure may proceed. Considering the intrinsic properties of GA, a number of approaches may be adopted. The following subsections provide a detailed account of the methods used and comparison have been made to iron out their differences.

B.2.a Encoding Mechanism. In order to proceed with GTW, the warping path must be encoded in the form of a chromosome. Consider that a warping path $\phi' = (\phi'_x, \phi'_y)$ is represented by a sequence of points in the (i_x, i_y) plane, i.e.,

$$\phi' = (\phi'_x(1), \phi'_y(1))\,(\phi'_x(2), \phi'_y(2)) \ldots (\phi'_x(T), \phi'_y(T)) \tag{5}$$

Eqn. (5) must satisfy the allowable region and local constraints. The order of points must move along the allowable paths which are restricted by the slope constraint. Thus, it is preferable to express the warping function ϕ by a sequence of sub-paths. The warping function ϕ can then be expressed as follows:

$$\phi = (p_1)(p_2) \ldots (p_L) \tag{6}$$

with initial points (i_0, i_0) and p_n being encoded as the allowable sub-paths.

B.2.b Fitness Function. Eqn. (3) is a distortion measurement between the two utterances X and Y. This provides the mechanism to evaluate the effectiveness of the warping function. However, the range of the distortion values calculated by Eqn. (3) can be unbounded. A fitness function is designed to normalize Eqn. (3) to a range from 0 to 1. The normalized value of Eqn. (3) is the fitness of the warping path function, which is used by the selection mechanism to evaluate the value of the "survival-of-the-fittest" of the warping function in subsequent generations. The procedures to calculate the fitness values are as follows:

1. Calculate distortion values d_n of each warping function in the population by Eqn. (3), i.e. $d_n = d_{\phi_n}(X, Y)$;
2. Find the maximum distortion value d_{max} in the population, i.e. $d_{max} = \max(d_1, d_2, \ldots, d_s)$;
3. Calculate the absolute differences dif_n between d_{max} and each d_n, i.e. $dif_n = d_{max} - d_n$;
4. Calculate the summation T of all differences calculated in step (3), i.e. $T = \sum dif_n$; and
5. Calculate fitness values f_n of each warping function by equation $f_n = dif_n/T$.

B.3 Genetic Time Warping with Relaxed Slope Weighting Function

To compare two utterance with large timing different is unrealistic. This problem can be alleviated by introducing a normalized slope weighting function on $m(k)$. This is possible when M_ϕ in Eqn. (3) is defined as:

$$M_\phi = \sum_{k=1}^{T} m(k) \tag{7}$$

The computation of M_ϕ can be clumsy for DTW particularly when $m(k)$ is varied dynamically. Whereas for the case in GTW, each path is coded as a chromosome, then the computation of M_ϕ presents no problem in the GTW formulation. With such an unique property, the definition of $m(k)$ is therefore relaxed and can be chosen arbitrarily. In this way, a GTW scheme with a relaxed slope weight function (GTW-RSW) can thus be performed.

B.4 Hybrid Genetic Algorithm

The GTW described above will produce results, and it is a well known fact that the classic GAs do have the inclination to search optimally over a wide range of the dynamic domain. However, they also suffer from being slow in convergence. To enhance this searching capability and improve the rate of convergence, problem-specific information is desirable for GA so that a hybrid-GA structure is formed. In the present hybrid-GA formulation, we add problem-specific knowledge to the crossover operator so that reproduction of offspring that possesses higher fitness values is realized.

In this hybrid-GTW scheme, the hybrid-crossover genetic operator is proposed. The hybrid-crossover operator is similar to the original crossover operator whose procedure is listed as follows:

1. Randomly select two chromosomes A and B and perform the normal crossover operation. An offspring C is produced;
2. Swap chromosomes A and B and perform the crossover procedures again. Another offspring D is produced; and
3. Instead of putting the offspring back into the population, a discrimination process is executed, such that the best chromosomes among A, B, C, and D will be put back into the population pool.

B.5 Parallel Genetic Algorithm

Parallel processing is an unique application for GA. In this case, there will be more chromosomes (warping paths) to be examined. However, the implementation of GA parallelism for GTW is not straight forward due to the need (in the selection steps) for a global statistics calculation which results in higher communication costs.

In order to reduce these costs, global selection is therefore avoided, but only the K-best solutions of each processor are considered for communication. The operational procedures are described as follows:

1. Each worker processor extracts K best solutions and $K \times N$ bad solutions from its population where N is the number of processors in the system provided that $K \times N <$ size of the population;
2. Each worker processor passes the K-best solutions to the farmer processor;
3. The farmer processor puts all received solutions into a group of $K \times N$ solutions and sends the group back to every processor;
4. Each worker processor replaces $K \times N$ bad solutions in its population by the received solutions.

A metric $|M_s - M_d|$ is provided to discriminating abilities for recognizing confused utterances, particularly for utterances that have similar acoustic properties, where M_s and M_d are the standard deviations of distortions of the same words and the standard deviations of different words, respectively. A lower value in $|M_s - M_d|$ implies that the ASR system has a poor ability to identify confused utterances, while a higher $|M_s - M_d|$ value implies that the ASR system has a high level of confidence in recognizing confused utterances. The results of $|M_s - M_d|$ for the five experiments are tabulated in Fig. 9.

It can be clearly shown in Fig. 9 that all the algorithms using the GA technique have higher values of $|M_s - M_d|$ than those of DTW, and therefore a higher discrimination ability than the conventional DTW.

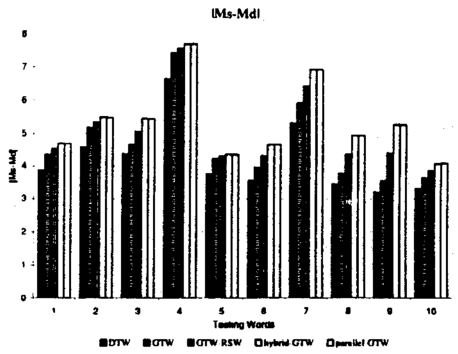


Fig. 9. The Chart of $|M_s - M_d|$ of Five Experiments

IV. HIERARCHICAL GA

Thus far, the concept of applying a GA to solve engineering problem is feasible and sound. However, despite the distinct advantages of a GA for solving complicated, constrained and multiobjective functions where other techniques may have failed, the full power of the GA in engineering application is yet to be exploited and explored.

To bring out the best use of GA, the knowledge of biological and medical is applied. It is known that the genetic structure of chromosome is formed by a number of gene variations, that are arranged in a hierarchical manner. Some genes dominate other genes and there are active and inactive genes in biological DNA. Such a phenomenon is a direct analogy to the topology of many engineering systems that have yet to be determined by an adequate methodology.

In light of this issue, a reprise of biological genetics was carried out. A method has been proposed to emulate the formulation of a biological DNA structure so that a precise hierarchical genetic structure can be formed for engineering purpose. The beauty of this concept is that the basic genetic computations are maintained. Hence, this specific genetic arrangement proves to be an immensely rich methodology for system modelling, and its potential uses in solving topological scientific and engineering problems should become a force to be reckoned with in future systems design.

A. Regulatory Sequences and Structural Genes

The biological background of the DNA structure has already been given in [12], [29]. A genetic product, generally known as a polypeptide, is produced by the DNA only when the DNA structure is experiencing biological and chemical processes. From such a genetic process, the genes can thus be classified into two different types:

- regulatory sequences (RSs) and
- structural genes (SGs)

The SGs are coded for polypeptides or ribonucleic acids (RNAs), while the RSs serve as the leaders that denote the beginning and ending of SGs, or participate in turning on or off the transcription of SGs, or function as initiation points for replication or recombination. One of the RSs found in DNA is called the promoter, and this activates or deactivates SGs due to the initialization of transcription. This initialization is governed by the Trans-acting Factor (TAF) [11] acting upon this sequence in the DNA. The transcription can only take place if a particular TAF is bound on the promoter. A polypeptide is then produced via a translation process in which the genetic message is coded in messenger RNA (mRNA). Therefore, a hierarchical structure is obtained within a DNA formation which is depicted in Fig. 10.

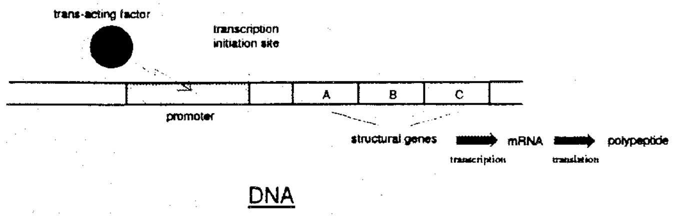


Fig. 10. Trans-acting Factor Bound on Promoter for the Initiation of Transcription

B. Active and Inactive Gene

Another surprising discovery in the founding of molecular biology was that active and inactive genes exist in the SGs. The active genes are separated into non-contiguous pieces along the parental DNA. The pieces that code mRNA are referred to as exons (active genes) and the non-coding pieces are referred as introns (inactive genes). During transcription, there is a process of splicing, Fig. 11 so that the final mRNA contains the exons only.

C. Hierarchical Chromosome Formulation

It is the existence of the inactive genes together with the commanding signals of TAF in promoter that ignite the innovated introduction of a hierarchical formulation of the chromosome for GA. This chromosome can be regarded as the DNA that has already been described, but consists of the parametric genes (analogy to structural genes in DNA) and the control genes (analogy to regulatory sequences in DNA). To generalize this architecture, a multiple level of control genes are introduced in a hierarchical fashion as illustrated in Fig. 12. In this case, the activation of the parametric gene is governed by the value of the first-level control gene, which is governed by the second-level control gene, and so on.

To indicate the activation of the control gene, an integer "1" is assigned for each control gene that is being ignited where "0" is for turning off. When "1" is signalled, the associated parameter genes due to that particular active control gene are activated in the lower level structure. It should be noticed that the inactive genes always exist within the chromosome even when "0" appears. This hierarchical architecture implies that the chromosome contains more information than that of the conventional GA structure. Hence, it is called Hierarchical Genetic Algorithm (HGA).

The use of the HGA is particularly important for determination of the structure or topology as well as the parametric optimization. Unlike the set-up of the conventional GA optimization, where the chromosome and the phenotype structure are assumed to be fixed or pre-defined, HGA operates without these constraints.

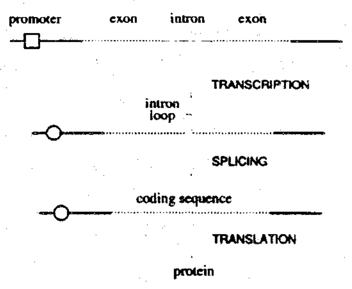


Fig. 11. Splicing

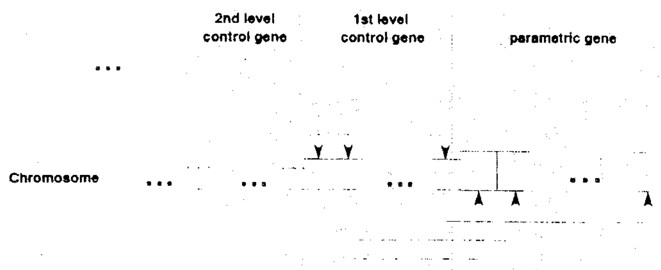


Fig. 12. Hierarchical Chromosome Structure

D. Genetic Operations

Since the chromosome structure of HGA is fixed, and this is true even for different parameter lengths, there is no extra effort required for reconfiguring the usual genetic operations. Therefore, the standard methods of mutation and crossover may apply independently to each level of genes or even for the whole chromosome if this is homogenous. However, the genetic operations that affect the high-level genes can result in changes within the active genes which eventually leads to a multiple change in the lower level genes. This is the precise reason why the HGA is not only able to obtain a good set of system parameters, but can also reach a minimized system topology.

E. Multiple Objective Approach

Multiple objective can be incorporated into HGA (MOHGA) which gives the flexibility of structure optimization. Since its main purpose is to determine the topology of the system, an extra objective function should be included for optimization. Based on the specific HGA chromosome structure, the topology information can be easily acquired from the control genes.

V. HGA Applications

A. *Robust* $\mathbf{H}^{\infty}$ *Control of a Benchmark Process*

In the case of robust control, the principal of the H^{∞} Loop Shaping Design Procedure (LSDP) [35] is used to design a pre-compensator (W_1) and a post-compensator (W_2) to shape the nominal plant (G) so that the singular values of the shaped plant ($G_s = W_2 G W_1$) has a desired open-loop shape. GA may be incorporated into the LSDP for searching the shaping function space in order to find a suitable robust controllers that an explicit close-loop performance is met.

Let's consider the benchmark plant model [19] that comprises the dominant dynamics and the high-frequency dynamics stated as:

$$G_{bp}(s) = G(s) \cdot G_{hf}(s) \qquad (8)$$

where the dominant dynamics, i.e. the nominal plant:

$$G(s) = \frac{K(-T_2 s + 1)\omega_0^2}{(s^2 + 2\zeta\omega_0 s + \omega_0^2)(T_1 s + 1)} \qquad (9)$$

while the high frequency dynamics:

$$G_{hf}(s) = \frac{\omega_\delta^2}{(s^2 + 2\zeta_\delta\omega_\delta s + \omega_\delta^2)(T_1^\delta s + 1)(T_2^\delta s + 1)} \qquad (10)$$

The high-frequency dynamics are fixed in time, taking the values of $T_1^\delta = 1/8$, $T_2^\delta = 1/12$, $\omega_\delta = 15$, and $\zeta_\delta = 0.6$ for all stress levels.

The dominant dynamics is time-varying in nature within the limited ranges: $T_1 = 5 \pm \Delta T_1$, $T_2 = 0.4 \pm \Delta T_2$, $\omega_0 = 5 \pm \Delta\omega_0$, $\zeta = 0.3 \pm \Delta\zeta$ and $K = 1 \pm \Delta K$, where the ranges of variation for different stress levels (S_1, S_2 and S_3) are tabulated in Table II.

TABLE II
Plant Parameters Variation at Three Stress Levels

	ΔT_1	ΔT_2	$\Delta\omega_0$	$\Delta\zeta$	ΔK
S_1	0.20	0.05	1.50	0.10	0
S_2	0.30	0.10	2.50	0.15	0.15
S_3	0.30	0.15	3.00	0.15	0.50

The requirements of the closed-loop performance due to a square wave oscillating reference signal with a strength between ±1.0 are listed as follows:

1. Plant output *must* be between ±1.5 *at all times*;
2. The overshoot and undershoot should preferably be less than 0.2 *most of the time*, though occasionally larger overshoots or undershoots are acceptable provided that the ±1.5 limits are preserved;
3. Control input to the plant saturates at ±5;
4. Fast settling time; and
5. Zero steady state tracking error (modulo high frequency noise).

According to the performance criteria of the benchmark, four performance indices, namely overshoot (ϕ_1), undershoot (ϕ_2), settling time (ϕ_3) and rising time (ϕ_4), are defined as:

$$\begin{aligned} \phi_1 &= \max(y) \\ \phi_2 &= -\min(y) \\ \phi_3 &= \max_t(|y - r| > 0.05) \\ \phi_4 &= \min_t(|y| \geq |0.8r|) \end{aligned} \qquad (11)$$

A.1 LSDP approach using HGA

The LSDP design [36] proposed that the weighting functions W_1, and/or W_2 can be chosen by evaluating the open-loop singular values of the plant, such that the weighted plant should possess an open-loop shape giving good closed-loop performance.

The schematic block diagram of the closed-loop system is depicted in Fig. 13. A criterion $\gamma_0 \leq \varepsilon_\gamma$ is used to determine the robustness of the controller K_s where ε_γ which is defined by the designer. A gain compensator K_p is used to ensure unity steady state gain for the tracking problem.

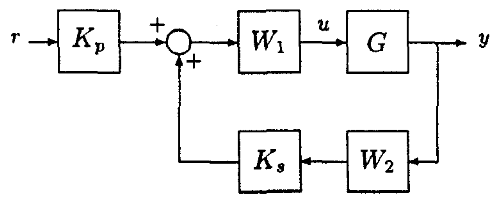


Fig. 13. The $\mathbf{H}^{\infty}$ control system

With the use of an HGA approach[48], it can search all the possible solutions within the specified maximum order. Optimization on the order and coefficients of W_1 and W_2 can be automatically determined by this approach. With the incorporating Pareto multi-objective scheme, which differs from the previous optimization scheme [48], this method also solves the soft and hard constraints problems that are posed in the design.

A.2 Chromosome Coding

HGA chromosome structure can be described as shown in Fig. 14 as an example to present the HGA chromosome arrangement for a transfer function $W(s)$, whose fundamental structure is

$$W(s) = g \times \frac{(s + b_1)(s + b_2)(s^2 + b_{11}s + b_{12})}{(s + a_1)(s + a_2)(s^2 + a_{11}s + a_{12})} \qquad (12)$$

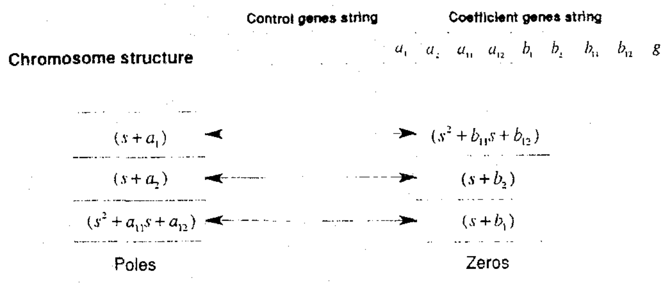


Fig. 14. Chromosome structure

The fundamental structures of W_1 and W_2 are defined as stated in Eqn. (12) excepted that a pure integrator is cascaded in W_1 to ensure zero steady error.

A.3 Multiobjective Approach

Since the benchmark problem is time-varying, it is impossible to design an optimal controller based only on the nominal plant. Hence, a class of extreme plants (totally q extreme plants) are defined. Objective functions are now indicated in Eqn. (13), as the largest (or worst) value of the performance indices in Eqn. (11). The detailed procedures can be referred to [49].

$$f_j = \max\{\phi_j(G_i, W_1, W_2)\} \quad \forall \quad i = 1, 2, \ldots, q; \quad j = 1, 2, 3, 4 \qquad (13)$$

A Pareto multi-objective preferable ranking method is adopted to handle such multi-objective problem. Chromosome I_1 is more preferable than chromosome I_2 if and only if

Condition I: $\gamma_0(I_2) \geq \varepsilon_\gamma$

$$\gamma_0(I_1) < \gamma_0(I_2)$$

Condition II: $\gamma_0(I_2) < \varepsilon_\gamma$ and $f_j(I_2) > 1.5$; $\forall j = 1, 2$

$$\begin{array}{lccc} \forall j = 1,2 & f_j(I_1) & \leq & f_j(I_2) \\ \exists k = 1,2 & f_k(I_1) & < & f_k(I_2) \end{array}$$

Condition III: $\gamma_0(I_2) < \varepsilon_\gamma$ and $f_2(I_2) \leq 1.5 < f_1(I_2)$

$$\begin{array}{lcl} f_1(I_1) & < & f_1(I_2) \\ f_2(I_1) & \leq & 1.5 \end{array}$$

Condition IV: $\gamma_0(I_2) < \varepsilon_\gamma$ and $f_1(I_2) \leq 1.5 < f_2(I_2)$

$$\begin{array}{lcl} f_2(I_1) & < & f_2(I_2) \\ f_1(I_1) & \leq & 1.5 \end{array}$$

Condition V: $\gamma_0(I_2) < \varepsilon_\gamma$ and $\forall j = 1, 2$ $f_j(I_2) \leq 1.5$

$$\begin{array}{lccc} \forall j = 1,2,3,4 & f_j(I_1) & \leq & f_j(I_2) \\ \exists k = 1,2,3,4 & f_k(I_1) & < & f_k(I_2) \end{array}$$

The above preferable comparison procedure accommodates the information of the hard constraints posed on $f_1(\leq 1.5)$, $f_2(\leq 1.5)$ and $\gamma_0(< \varepsilon_\gamma)$, together with the soft constraints for f_3 and f_4. Degradation in f_1 (or f_2) within the specified bound is acceptable if improvement of f_2 (or f_1) exceeds the specified bound is achieved. The chromosome I can then be ranked as

$$rank(I) = 1 + p \qquad (14)$$

if I is preferred by other p chromosomes in the population. Fitness is then assigned to the chromosome according to its rank in the population.

A.4 Results

To demonstrate the effectiveness of this approach, eight extreme systems were selected and examined. A population of size 40 is randomly generated initially. At each MO-HGA cycle, eight offspring are generated through crossover and mutation. After 1000 iterations, the final weighting functions W_1 and W_2 are obtained as indicated in Table III.

TABLE III
RESULTED WEIGHTING FUNCTIONS FOR THREE STRESS LEVELS

	W_1	W_2
S_1	$\frac{7.80(s+0.13)}{s(s+1.74)}$	1
S_2	$\frac{3.90(s+0.18)}{s(s+1.11)}$	1
S_3	$\frac{9.27(s^2+0.52s+0.70)}{s(s^2+1.84s+0.60)}$	$\frac{s+0.89}{(s+1.53)(s+1.98)}$

Work proceeded to justify the closed loop system performance. This was performed repeatedly 15 times in each of the three stress levels, so that a much clearer picture of the plant dynamics variation could be obtained. The simulated results obtained for each stress level are shown in Figs. 15–17 respectively. The results indicate that the design specifications have been achieved despite the system variations in different stress levels.

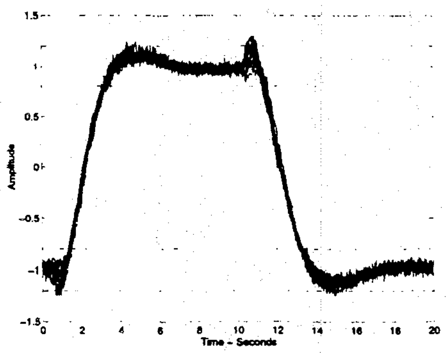


Fig. 15. Plant Response to Square Reference in Stress Level 1

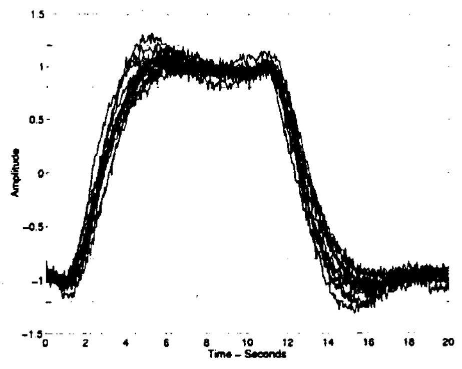


Fig. 16. Plant Response to Square Reference in Stress Level 2

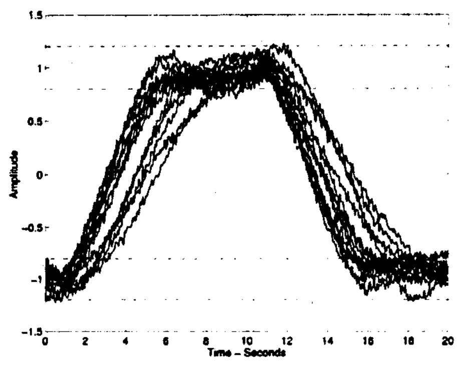


Fig. 17. Plant Response to Square Reference in Stress Level 3

By comparison, the MO-HGA design approach based on the H^∞ LSDP formulation provides an even better simplicity versus performance which benefits the small to moderate plant variations [19]. Furthermore, to the contrary of robust H^∞ design [52], the present methods of controlling the plant dynamics in stress level 3 (S_3) are also much improved and those of the obtained controller are considerably simplified.

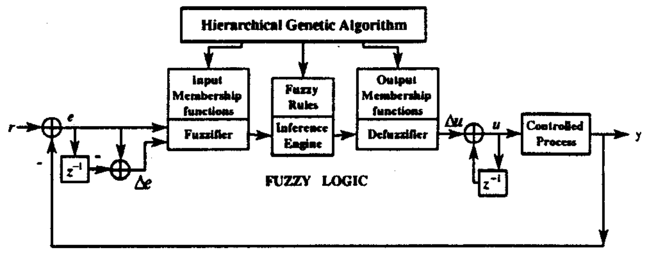
B. Fuzzy Logic System

The basic procedures of designing a FLC has been well established [10]. The operating procedures for these variables are usually done manually but this often yields a sub-optimal performance despite some other automatic tuning schemes [27].

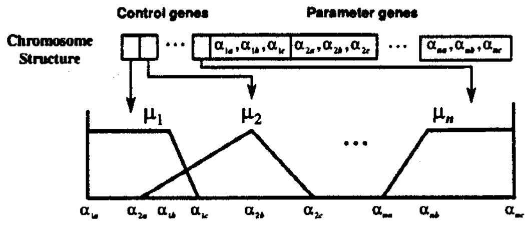
Considering that the main attribute of the HGA is its ability to solve the topological structure of an unknown system, then the problem of determining the fuzzy membership functions and rules could also fall into this category. This approach has a number of advantages:

- an optimal and the least number of membership functions and rules are obtained;
- no pre-fixed fuzzy structure is necessary;
- simpler implementing procedures and less cost are involved; and
- meets design criteria that can be multi-objective and constrained

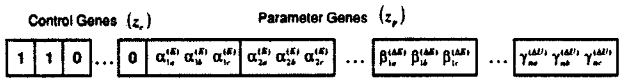
The conceptual idea is to have an automatic and intelligent scheme to tune the fuzzy membership functions and rules, in which the closed loop fuzzy control strategy remains unchanged, as indicated in Fig. 18a. In this system, Mamdani's approach [31] is applied for generating the fuzzy output value. The crisp value is then calculated by the centroid defuzzication method.



(a) HGA Fuzzy Logic Control System



(b) Hierarchical Membership Chromosome



(c) Final Hierarchical Chromosome

Fig. 18. Formulation of HGA Fuzzy Logic Control System

The chromosome of a particular fuzzy set is shown in Fig. 18b. The chromosome consists of the usual two types of genes, the control genes and parameter genes. The control genes, in the form of bits, determine the membership function activation, whereas the parameter genes are in the form of real numbers to represent the membership functions.

With the two input fuzzy sets of error signals (e) and error rate (Δe), and the output fuzzy set of Δu, we can thus construct the overall membership chromosome structure as in Fig. 18c.

The parameter genes (z_p) of the membership chromosome take the form:

$$\begin{aligned} z_p = & \Big\{\alpha_{1a}^{(E)}, \alpha_{1b}^{(E)}, \alpha_{1c}^{(E)}, \cdots, \alpha_{ma}^{(E)}, \alpha_{mb}^{(E)}, \alpha_{mc}^{(E)}, \\ & \beta_{1a}^{(\Delta E)}, \beta_{1b}^{(\Delta E)}, \beta_{1c}^{(\Delta E)}, \cdots, \beta_{na}^{(\Delta E)}, \beta_{nb}^{(\Delta E)}, \beta_{nc}^{(\Delta E)}, \\ & \gamma_{1a}^{(\Delta U)}, \gamma_{1b}^{(\Delta U)}, \gamma_{1c}^{(\Delta U)}, \cdots, \gamma_{pa}^{(\Delta U)}, \gamma_{pb}^{(\Delta U)}, \gamma_{pc}^{(\Delta U)}\Big\} \end{aligned}$$

where m, n and p are the maximum allowable numbers of fuzzy subset of E, ΔE and ΔU, respectively; $\alpha_{ia}^{(E)}, \alpha_{ib}^{(E)}, \alpha_{ic}^{(E)}$ define the input membership function of i-th fuzzy subset of E; $\beta_{ja}^{(\Delta E)}, \beta_{jb}^{(\Delta E)}, \beta_{jc}^{(\Delta E)}$ define the input membership function of j-th fuzzy subset of ΔE; and $\gamma_{ka}^{(\Delta U)}, \gamma_{kb}^{(\Delta U)}, \gamma_{kc}^{(\Delta U)}$ define the output membership function of k-th fuzzy subset of ΔU.

Considering that there are various types of gene structure, a number of different genetic operations have been designed. For the crossover operation, an one-point crossover is applied separately for both the control and parameter genes of the membership chromosomes within certain operation rates.

Bit mutation is applied for the control genes of the membership chromosome. Each bit of the control gene is flipped ("1" or "0") if a probability test is satisfied (a randomly generated number, r_c, is smaller than a pre-defined rate). As for the parameter genes, which are real-number represented, random mutation is applied.

The genetic operations on membership chromosome may destroy the required order of the membership function ($\alpha_{ia}^{(E)} \leq \alpha_{ib}^{(E)} \leq \alpha_{ic}^{(E)}$). Hence the parameters are sorted to ensure the validity of the membership chromosome. To be able to complete the FLC design, the evolution of fuzzy rules and fitness evaluation should be determined. Reader may consult [50] for details.

B.1 Experimental Results

To test the design of the HGA fuzzy logic controller, an experimental piece of equipment which consisted of a water pump having a 1.5 horse power engine and a water tank was used for the investigation. It simulated a constant pressure booster pump system that is designed for a water supply system [22].

The compensated actuator unit was the water pump with a variable frequency converter (VFC) attached. The actuating signal came from a pressure sensor placed in a pipe downstream and its output signal was fed back into the VFC to change the pump speed.

The underlying problem of the control is the nonlinear characteristics for both QH characteristics of the actuating power source (pump speed) and the dissipated energy due to the pipe characteristics. When the control loop is closed, the dissipated energy of the pipe is to be suppressed adequately by the actuating power so as to keep the variation of the perturbed pressure at a minimum level, and yet the water flow rate is allowed to changed freely to suit the demand.

B.2 Fuzzy Learning and Optimization

The design specification is to meet two objective functions:

- Minimum output steady state error
- No overshoot and undershoot are allowed.

The fitness function is hence defined as

$$f = \begin{cases} g & \text{No overshoot or undershoot} \\ g + \zeta & \text{else} \end{cases} \qquad (15)$$

where g is the mean steady state error and ζ is a penalty value when there is overshoot or undershoot.

The obtained membership functions and the corresponding rule table are shown in Fig. 19 and Table IV, respectively.

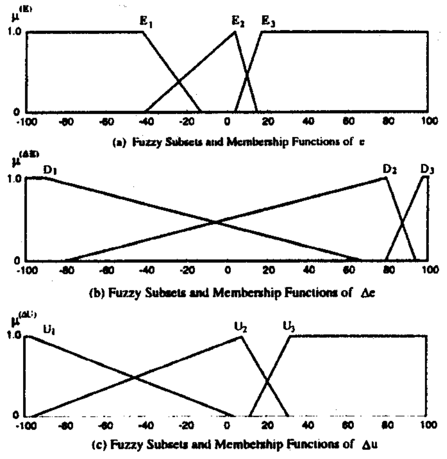


Fig. 19. Final Fuzzy Subsets and Membership Function for Water Pump Control System

TABLE IV
Optimal Rule Table

		Error Rate Fuzzy Set		
		D_1	D_2	D_3
Error	E_1	U_1	U_2	U_2
Fuzzy	E_2	U_2	U_2	U_3
Set	E_3	U_2	U_3	U_3

It has been shown that a reduced size of subsets of fuzzy membership functions and rules is obtained by the use of HGA while still meeting the requirements of system performance. This result is considered to be compatible to those obtained using the conventional fuzzy logic design schemes. The final results are shown in Fig. 20 and Fig. 21.

C. *Neural Network*

A general neural network topology taking the form of a multilayer feedforward network is depicted in Fig. 22.

The basic processing element in NN is called a neuron. A neuron consists of an activity level, a set of input and output connections with a bias value associated to each connection.

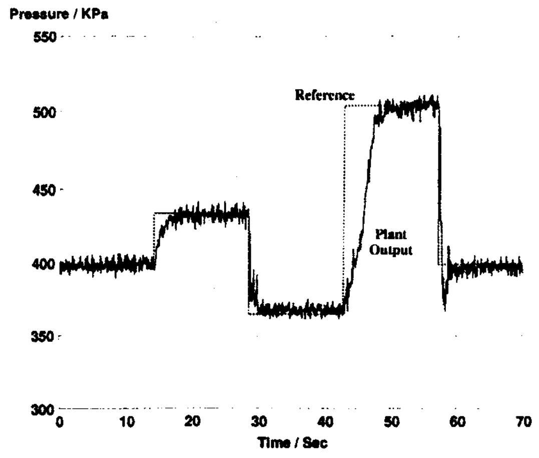


Fig. 20. Set-point Following

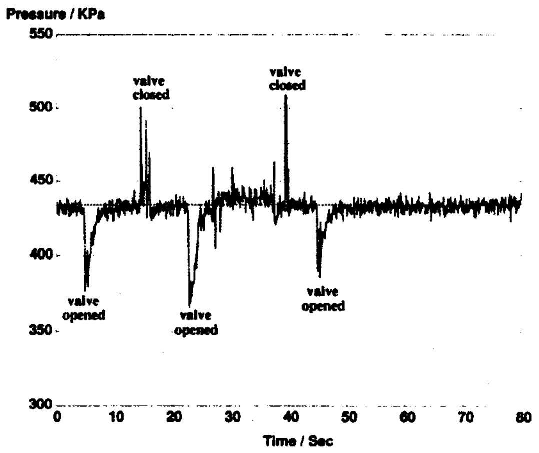


Fig. 21. Closed Loop Disturbance

The output of the neuron is determined as:

$$y = f\left(\sum_{i=1}^{n} \omega_i x_i + \tau\right) \tag{16}$$

where $x_1, x_2, \ldots, x_n$ are input signals; $\omega_1, \omega_2, \ldots, \omega_n$ are connection weightings; τ is the bias value; and f is a defined output function that may be a sigmoid, tanh, step function etc.

In order for this topology to function according to the design criteria, a learning algorithm that is capable of modifying the network parameters, as indicated in Eqn. (16), is of paramount important to the NN. The backpropagation (BP) technique, that employs a gradient descent learning algorithm [43], is commonly used by the NN community. This approach requires a pre-defined topology which implies that the numbers of neurons and connections must be known a prior. Furthermore, as the network complexity increases, the performance of BP decreases rapidly. The other deficit of BP is its use of gradient search algorithms, where discontinuous connection weightings cannot be handled.

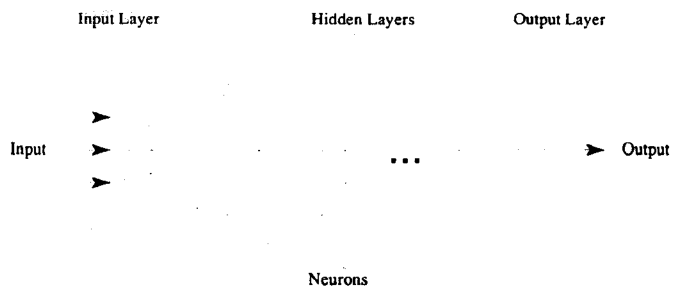


Fig. 22. A Multilayer Feedforward Neural Network Topology

Having realized the pros and cons of NN, the bottle-neck problem lies within the optimization procedures that are implemented to obtain an optimal NN topology. Hence, the formulation of the HGA is well suited for this purpose.

Fig. 23 shows the chromosome representation in the HGANN system. The control genes in the form of bits, are the genes for layers and neurons activation. The connection genes, in real-value representation, are the genes for connection weightings and neuron bias. A NN defined by this structure is depicted in Fig. 23.

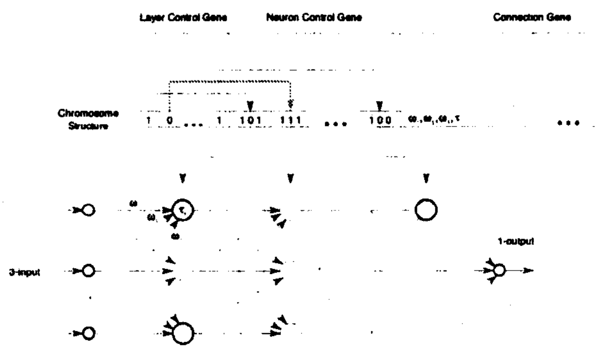


Fig. 23. HGANN Chromosome Structure

To formulate such an HGANN, its overall system block diagram is shown in Fig. 24. The HGANN has the ability to learn the network topology and the associated weighting connection concurrently. Each learning cycle is known as a generation.

C.1 Fitness Functions

The objective of training the network is to minimize two different parameters: the accuracy of the network (f_1) and the complexity of the network (f_2). This complexity (f_2) is simply defined by the number of active connections in the network. It is calculated based upon the summation of the total number of active connections taking place. The accuracy of the network (f_1) is defined as:

$$f_1 = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{17}$$

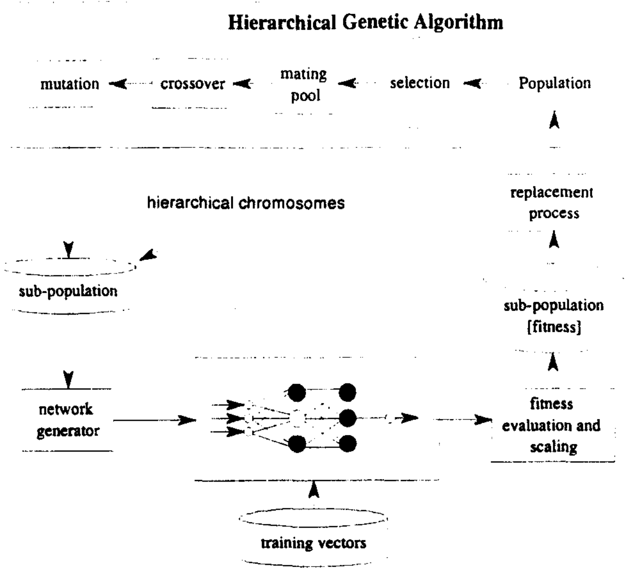


Fig. 24. Block Diagram of the Overall HGANN Operation

where N is the size of the training vectors; $\hat{y}_i$ and y_i are the network output and desired output for the i-th pattern of the training vector, respectively.

Since there are two different objective functions, (f_1) and (f_2) of the network optimization process, the fitness value of chromosome z is thus determined:

$$f(z) = \alpha \cdot rank[f_1(z)] + \beta \cdot f_2(z) \tag{18}$$

where α is accuracy weighting coefficient; β is complexity weighting coefficient; and $rank[f_1(z)] \in Z^+$ is the rank of z in the population. In general, $\alpha > \beta \cdot M$ [46] where M is the maximum active number of connections in the NN.

The selection rate of a chromosome z, *tsr(z)*, is determined by:

$$tsr(z) = \frac{F - f(z)}{(size[P^{(k)}] - 1) \cdot F} \tag{19}$$

where F is the sum of the fitness value of all chromosomes.

C.2 Experimental Results

To verify the performance of the proposed HGANN system, a real-life application is introduced. A medical expert system is to be developed for the diagnoisis of breast cancer to classify a tumor as either being benign or malignant.

The original data was obtained from the University of Wisconsin Hospitals, Madison [34]. There are nine input attributes includes clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses, normalized to $[0,1]$.

The 699 cases are split into two portions, 100 cases as training data and 599 case as testing data. The idea is that the performance of a network on the testing set estimates its performance for real use. It turns out that there is absolutely no information about the testing set examples or the testing set performance of the network available during the training process.

To proceed with the HGANN formulation, the fundamental network consists of nine-input and one-output with four hidden layers, similar to Fig. 23. In addition to the parametric genes representing the weightings, the chromosome has two types of control genes, neuron control genes and layer control genes, dedicated for the activation of neurons and layers, respectively. The "1" signifies the output as being benign, and vice verse.

A greatly simplified network with only two neurons (one hidden layer) is finally obtained. This final networks obtained by HGANN are then applied to the testing data. The classification accuracy of the network generated by HGANN is about 96.33%. It is a slightly better than the result obtained from BP (95%) and also other reported results (93.5% [56] and 93.7% [58]).

VI. Conclusion

An attempt to outline the features of GA in terms of the genetic functionality of operators, the problem formulation, the inherent capability of GA for solving complex and conflicting problems as well as its various practice applications is given in this tutorial. The theme is oriented from industrial application basis. The purpose is to introduce this emerging technology to engineers for whom have little or even zero knowledge in GA. It is also reasonably to believe that this tutorial should have comprised enough materials to encourage, and to arouse the interest to newcomers so that GA may be implemented for their practice problems.

References

[1] J.E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 101-111.

[2] J.E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 14–21.

[3] S. Baluja, "Structure and performance of fine-grain parallelism in genetic search," in *Proc. 5th Int. Conf. Genetic Algorithm*, 1993, .

[4] L. Booker, "Improving search in genetic algorithms," in *Genetic Algorithms and Stimulated Annealing*, L. Davis (Eds), 1987, pp. 61–73.

[5] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," *IlliGAL Report No. 95007*, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, July 1995.

[6] Y. Davidor, "A genetic algorithm applied to robot trajectory generation," in *Handbook of Genetic Algorithms*, L. Davis (Eds), 1991, pp. 144–165.

[7] L. Davis, "Job shop scheduling with genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 136–140.

[8] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 61–69.

[9] L. Davis, *Handbook of genetic algorithms*, Van Nostrand Reinhold, 1991.

[10] Driankov, M., "An Introduction of Fuzzy Control," *Nature*, Springer-Verlag, 1993.

[11] W.S. Dyann and R. Tjian, "Control of eukaryoti messenger RNA synthesis by sequence-specific DNA-binding proteins," *Nature*, vol. 316, 1985, pp. 774–778.

[12] A.E.H. Emery and D.L. Rimoin, *Principles and practice of medical genetics*, 2nd Ed., Churchill, Liveringstone, 1990.

[13] L.J. Eshelman, R. Caruna, and J.D. Schaffer, "Biases in the crossover landscape," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 10–19.

[14] C.M. Fonseca and P.J. Fleming, "Genetic algorithms for multiobjecitve optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 416–423.

[15] C.M. Fonseca and P.J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Research Report No. 527*, Dept. of Automatic Control and Systems Eng., University of Sheffield, UK.

[16] D.E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization", in *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 41-49.

[17] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, 1989.

[18] D.E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and block," *Technical Report No. 90001*, University of Illinois, Sept 1990.

[19] S. F. Graebe, "Robust and Adaptive Control of an Unknown Plant: A Benchmark of New Format," *Automatica*, vol. 30, no. 4, pp. 567–575, Apr 1994.

[20] J.J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans Systems, Man, and Cybernetics*, vol. 16, no. 1, Jan/Feb 1986, pp. 122–128.

[21] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural Optimization*, vol. 4, 1992, pp. 99–107.

[22] Y.C. Ho, K.F. Man, K.P. Cheuk and K.T. Ng, "A Fully Automated Water Supply System for High Rise Building," *Proc. 1st Asian Control Conference*, pp. 1–4, Toyko, Japan, 1994.

[23] J.H. Holland, *Adaptation in natural and artificial systems*, MIT Press, 1975.

[24] R.B. Hollstien, "Artificial genetic adaptation in computer control systems," *PhD thesis*, University of Michigan, 1971.

[25] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm," *IlliGAL Report 93005*, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA.

[26] C.Z. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representations in genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 31–36.

[27] Jang, J.-S.R. and C.-T. Sun (1995): Neuro-fuzzy modeling and control. Proc. IEEE, **83(3)**, pp. 378–406.

[28] J. Koza, "Evolution and co-evolution of computer programs to control independently-acting agents," *Animals to Animals*, J.A. Ineger and S.W. Willson (Eds.) Cambridge, MA: MIT Press/Bradford Books, 1991.

[29] A.L. Lehninger, D.L. Nelson and M.M. Cox, *Principles of biochemistry*, 2nd Edition, 1993.

[30] S.W. Mahfoud, "Population sizing for sharing methods," *IlliGAL Report No. 94005*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Aug 1994.

[31] E.H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *Int. J. Man-Machine Studies*, vol. 7, pp. 1–13, 1975.

[32] K.F. Man, K.S. Tang and S. Kwong, "Genetic algorithms: Concepts and applications," *IEEE Trans. on Industrial Electronics*, vol. 43, no. 5, Oct 1996, pp. 519–534.

[33] K.F. Man, K.S. Tang, S. Kwong and W.A. Halang, *Genetic algorithms for control and signal processing*, Springer-Verlag, London, 1997.

[34] O.L. Mangasarian and W.H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, no. 5, pp. 1–18, 1990.

[35] D. McFarlane and K. Glover, "Robust controller design using normalized coprime factor plant descriptions," *Information and Control Sciences*, Springer-Verlag, 1990.

[36] D. McFarlane and K. Glover, "A Loop Shaping Design Procedure Using $\mathbf{H}^\infty$ Synthesis," *IEEE Trans. Automatic Control*, vol. AC–37, no. 6, pp. 749–769, 1992.

[37] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program*, 3nd Ed., Springer-Verlag, 1996.

[38] L.R. Rabiner and B.H. Juang, *Fundamentals of speech recognition*, Prentice-Hall, 1994.

[39] A.E. Rosenberg, "Evaluation of an automatic speaker verification system over telephone line," *Bell Syst. J.* vol. 55, pp. 723–744, 1976.

[40] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 26, 1978, pp. 43–49.

[41] J.D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithm*, 1985, pp. 93–100.

[42] W.M. Spears and K. DeJong, "An analysis of multi-point crossover," *Foundations of Genetic Algorithms*, G.J.E. Rawlins (Eds), 1991, pp. 301–315.

[43] P.K. Simpson, *Artificial neural systems: Foundations, paradigms, applications, and implementations*, Pergamon Press, pp. 100–135, 1990.

[44] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 2-9.

[45] G. Syswerda, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms*, 1991, pp. 332–349.

[46] K.S. Tang, C.Y. Chan, K.F. Man and S. Kwong, "Genetic structure for NN topology and weights optimization," *Proc 1st IEE/IEEE Int. Conf. on GAs in Engineering Systems: Innovations and Applications*, Sheffield, UK, 1995, pp. 250-255.

[47] K.S. Tang, K.F. Man, S. Kwong, C.Y. Chan and C.Y. Chu "Application of the genetic algorithm to real-time active noise control," *Journal of Real-Time Systems*, vol. 11, no. 3, Nov 1996, pp. 289-302.

[48] K.S. Tang, K.F. Man, D.W. Gu, "Structured genetic algorithm for robust H_∞ control system design," *IEEE Trans. Industrial Electronics*, vol. 43, no. 5, Oct 1996, pp. 575-582.

[49] K.S. Tang, K.F. Man and J.Y. Ke, "Hierarchical genetic algorithm for robust H_∞ control of a benchmark process," in *Proc. Asian Control Conf.*, Korean, July 1997.

[50] K.S. Tang, K.F. Man, Z.F. Liu and S. Kwong, "Minimal fuzzy memberships and rules using hierarchical genetic algorithms," *IEEE Trans. on Industrial Electronics*, (accepted).

[51] R. Tanse, "Distributed genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 434-439.

[52] J. F. Whidborne, G. Murad and D-W Gu. and I. Postlethwaite, "Robust Control of an Unknown Plant — the IFAC 93 Benchmark" *Int. J. Control*, vol. 61, no. 3, pp. 589–640, 1995.

[53] D. Whitley, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 116–121.

[54] D. Whitley, "A genetic algorithm tutorial," *Technical Report CS-93-103*, Department of Computer Science, Colorado State University, Nov 1993.

[55] D. Wienke, C. Lucasius and G. Kateman, "Multicriteria target vector optimization of analytical procedures using a genetic algorithm. Part I. Theory, numerical simulations and application to atomic emission spectroscopy," *Analytica Chimica Acta*, vol. 265, no. 2, 1992, pp. 211–225.

[56] W.H. Wolberg and O.L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. of the National Academy of Sciences*, vol. 87, pp. 9193–9196, 1990.

[57] A.H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*, J.E. Rawlins (Ed.), Morgan Kaufmann, 1991, pp. 205-218.

[58] J. Zhang, "Selecting typical instances in instance-based learning," *Proc. 9th Int. Conf. Machine Learning* , pp. 470–479, 1992.