Genetic Algorithms: Concepts and Applications

K. F. Man, *Member, IEEE,* K. S. Tang, and S. Kwong, *Member, IEEE*

Abstract—**This paper introduces genetic algorithms (GA) as a complete entity, in which knowledge of this emerging technology can be integrated together to form the framework of a design tool for industrial engineers. An attempt has also been made to explain "why" and "when" GA should be used as an optimization tool.**

I. Introduction

THE USE of genetic algorithms (GA) for problem solving is not new. The pioneering work of J. H. Holland in the 1970's proved to be a significant contribution for scientific and engineering applications. Since then, the output of research work in this field has grown exponentially although the contributions have been, and are largely initiated, from academic institutions world-wide. It is only very recently that we have been able to acquire some material that comes from industry. The concept of this is somehow not clearly understood. However, the obvious obstacle that may drive engineers away from using GA is the difficulty of speeding up the computational process, as well as the intrinsic nature of randomness that leads to a problem of performance assurance.

Nevertheless, GA development has now reached a stage of maturity, thanks to the effort made in the last few years by academics and engineers all over the world. It has blossomed rapidly due to the easy availability of low-cost but fast-speed small computers. Those problems once considered to be "hard" or even "impossible," in the past are no longer a problem as far as computation is concerned. Therefore, complex and conflicting problems that require simultaneous solutions, which in the past were considered deadlocked problems, can now be obtained with GA.

Furthermore, the GA is not considered a mathematically guided algorithm. The optima obtained is evolved from generation to generation without stringent mathematical formulation such as the traditional gradient-type of optimizing procedure. In fact, GA is much different in that context. It is merely a stochastic, discrete event and a nonlinear process. The obtained optima is an end product containing the best elements of previous generations where the attributes of a stronger individual tend to be carried forward into the following generation. The rule of the game is "survival of the fittest will win."

In this sphere, there is an endless supply of literature describing the use of GA. The sheer number of references quoted in this paper is an apt indicator of the extensive work being done in this domain. This does not include the index finding from [1]. The technical knowledge of "what" GA is and "how" it works are well reported. This paper tries not to cover the same ground. Rather, there is room for the introduction of GA as a complete entity, in which knowledge of this emerging technology can be integrated together to form the framework of a design tool for industrial engineers. Moreover, a brave attempt has also been made to explain "why" and "when" we should use GA as an optimization tool. It is anticipated that there is sufficient materials being generated in this paper to support this claim.

This paper starts by giving a simple example of GA, as described in Section II, in which the basic framework of GA is outlined. This example forms the cornerstone to the architecture of this paper. For the benefit of newcomers to this particular field, the essential schema theory and building block hypothesis of genetic algorithms are briefly given in Section III. What makes GA work and how does it improve its evolution are the essence of GA. There are a number of variations used to achieve these tasks, and each task has its own merit. In Section IV, a range of structural modifications for GA in order to improve its performance are thus recommended.

Since so much has already been published about what can be done with GA, a short list of items cataloguing the advantages of using GA is given in Section V. In Section VI, an account of what GA "cannot do" is given. The well-known phenomena of deception and genetic drift are described. In addition, the problems concerning the real time and adaptiveness of GA are also reported.

As this paper is targeted at a specific audience, the collection of practical systems being implemented are introduced in Section VII, whereas Section VIII outlines the possibility of integrating GA into emerging technologies such as neural networks and fuzzy systems. Finally, the conclusions reached in Section XI and recommendations for future works are also given.

II. Basic Concepts of Genetic Algorithms

The basic principles of GA were first proposed by Holland [66]. Thereafter, a series of literature [33], [52], [89] and reports [10], [11], [102], [118] became available. GA is inspired by the mechanism of natural selection, a biological process in which stronger individuals are likely be the winners in a competing environment. Here, GA uses a direct analogy of such natural evolution. It presumes that the potential solution of a problem is an individual and can be represented by a set of parameters. These parameters are 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 solving the problem, and this value is closely related to its objective value.

Manuscript received September 21, 1995; revised November 26, 1995.
The authors are with the City University of Hong Kong, Hong Kong.
Publisher Item Identifier S 0278-0046(96)03295-9.

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 total fitness N
- Return the first population member whose fitness added to the fitness of the preceding population members, is greater than or equal to n.

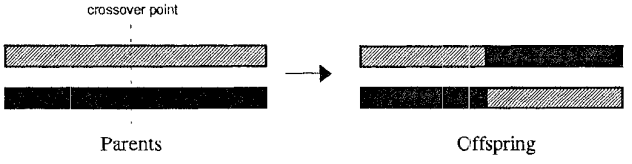


Fig. 1. Example of one-point crossover.

Throughout a genetic evolution, a 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 [83]. In each cycle of genetic operation, termed an evolving process, a subsequent generation is created from the chromosomes in the current population. This can only be successful 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 [33] is one of the most commonly used techniques in such a proportionate selection mechanism. To illustrate this further, the selection procedure is listed in Table I.

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), the amount of variation of individuals between different generations, or a predefined value of fitness.

In order to facilitate the GA evolution cycle, two fundamental operators—crossover and mutation—are required, although the selection routine can be termed as the other operator. To further illustrate the operational procedure, a 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 a typical value of between 0.6–1.0 is normally used as the probability of crossover.

However, for mutation, the process is applied to each offspring individually after the crossover exercise. It alters each bit randomly with a small probability (p_m) with a typical value of less than 0.1.

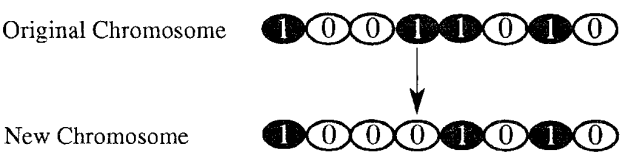


Fig. 2. Bit mutation on the fourth bit.

```
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.

The choice of p_m, p_c as the control parameters can be a complex, nonlinear optimization problem. 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 [36] and [59]:

- For large population size (100)
  crossover rate, p_c =0.6
  mutation rate, p_m = 0.001
- For small population size (30)
  crossover rate, p_c = 0.9
  mutation rate, p_m = 0.01

Fig. 3 summarizes the standard genetic algorithm.

A. Example of GA for Optimization

There is no better way to show how GA works than by going through a real, but simple, example to demonstrate its effectiveness.

1) Problem: To search the global maximum point of the following objective function (see Fig. 4):

$$z = f(x, y) \tag{1}$$

where $x, y \in [-1, 1]$.

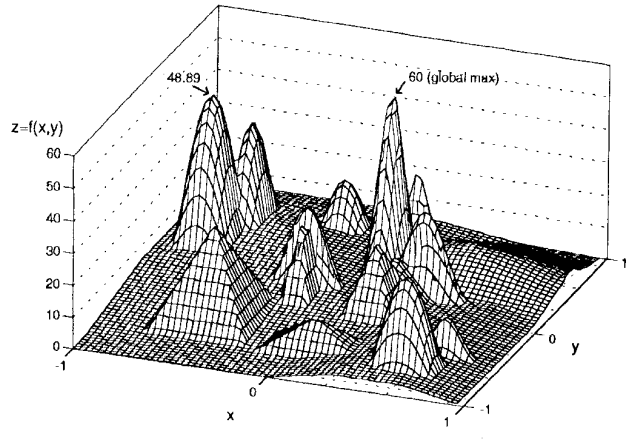


Fig. 4. A multimodal problem.

2) Implementation: The chromosome is to be formed by a 16-bit binary string representing x, and the y coordinates with 8-bit resolution each. One-point crossover and bit mutation are applied with operation rates of 0.85 and 0.1, respectively. Population size is set to 4 (In general, this number should be much larger). Here, only two offspring are generated for each evolution cycle.

A number of software packages can be utilized for this exercise, see Appendix. In this example, it is conducted via simulation based on MATLAB with the Genetic Toolbox [19]. Fig. 5 shows the typical genetic operations and the changes of the population from first generation to second generation. Fig. 6 clearly demonstrates that GA is capable of escaping from local maxima to find the global maximum point.

III. Theory and Hypothesis

Having established the fundamental principles of GA in structure arrangement and operational procedure, we can now carry on to gain a deeper understanding of GA. Thus far, the necessary justification of how GA works is yet to be illustrated. On this front, there are two schools of thoughts for explanation: schema theory and building block hypothesis.

A. Schema Theory

> *Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm* [89].

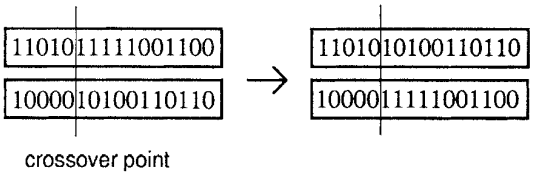
The design methodology of GA relies heavily on Holland's notion of schemata. It simply states that, schemata are sets of strings (encoded form of the chromosome) that have one or more features in common. A schema is built by introducing a "don't care" symbol "#" into the alphabet of genes, e.g., #1101#0. A schema represents all strings (a hyperplane or subset of the search space), which match it on all positions other than "#." It is clear that every schema matches exactly 2^r strings, when "r" is the number of don't care symbols "#" in the schema template. For example, the set of the schema #1101#0 is $\{1110110, 1110100, 0110110, 0110100\}$.

Let $\zeta(S, t)$ be the number of strings matched by schema "S" in the tth generation; $\delta(S)$ be the *defining length* of the schema S which is defined as the distance between the outermost fixed position; $o(\cdot)$ be the *order* of schema which is the

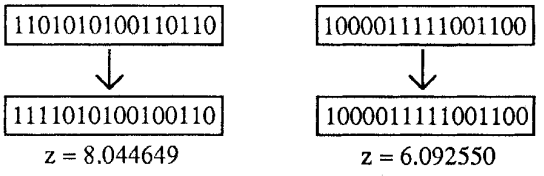
STEP 1: Parent Selection

First Population	Objective Value z = f(x,y)
1100110110101000	3.481746
0101010110110101	3.668023
→ 1000010100110110	6.261380
→ 1101011111001100	12.864222

STEP 2: CROSSOVER



STEP 3: MUTATION



STEP 4: Reinsertion

Second Population	Objective Value z = f(x,y)
→ 1111010100100110	8.044649
→ 1000011111001100	6.092550
1000010100110110	6.261380
1101011111001100	12.864222

Fig. 5. Generation to generation.

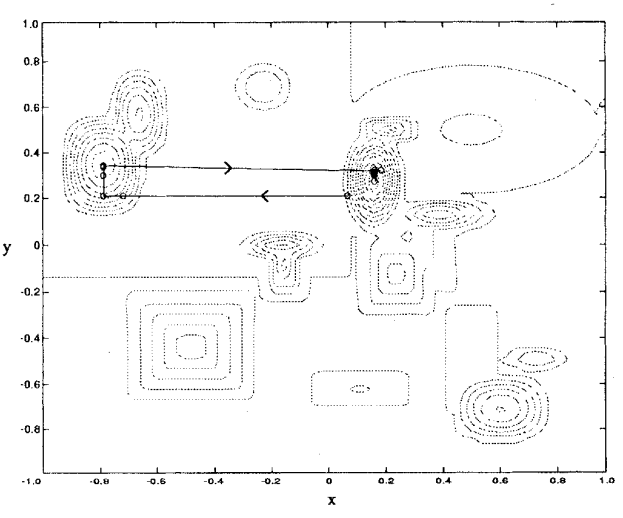


Fig. 6. An example of GA.

number of fixed positions present in the schema; $f(S, t)$ be the fitness value of schema S defining as the average fitness of all strings in the population matched by the schema S; L be the length of the chromosome; and $F(t)$ be the average number of occurrences of S.

Taking the effects of proportionate selection, of one-point crossover and mutation into account, a reproductive schema growth equation is thus obtained and expressed in (2). A detailed derivation of this equation can be referred to [89].

$$\zeta(S, t+1) \geq \zeta(S, t) \frac{f(S,t)}{F(t)} \cdot \left[1 - p_c \frac{\delta(S)}{L-1} - o(S)p_m\right] \quad (2)$$

where p_c, p_m are the operation rate of crossover and mutation, respectively.

The *implicit parallelism lower bound* derived by Holland provides a figure about the number of schemata which are processed in a single cycle in the order of N^3, where N is the population size.

Despite this formulation, it has limitations that lead to the restriction of its use. First, the predictions of the GA could be useless or misleading on some problems [63]. Secondly, the value of $f(S, t)$ in the current population may differ significantly from the value of $f(S, t)$ in the next, since schema can interfere with one other. Thus, using the average fitness is only relevant to the first population [60]. After this, the sampling of strings will become biased and the inexactness makes it impossible to predict computational behavior.

B. Building Block Hypothesis

> *A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building block* [89].

The genetic operators we normally refer to crossover and mutation have the ability to generate, promote, and juxtapose (side by side) building blocks to form the optimal strings. Crossover tends to conserve the genetic information present in the strings. Thus, when the strings for crossover are similar, its capacity to generate new building blocks diminishes. Mutation, however, is not a conservative operator but capable of generating new building blocks radically.

In addition, parent selection is an important procedure to be devised. It tends to bias toward building blocks with higher fitness values, and at the end ensures their representation from generation to generation.

IV. Structure Modification of Genetic Algorithms

Because of the GA mechanism, which is neither governed by differential equations nor behaves like a continuous function, it is therefore not difficult to conceive that a simple structure of GA can be devised for many practical applications as an optimizer. It is also without doubt that a standard GA is capable of solving a difficult problem which the conventional gradient-based or hill-climbing techniques might have trouble with, even to the extent of causing a complete failure or breakdown in computation.

However, a standard GA has many limitations. As the problem becomes complicated, multitasking, and conflicting, the structure of a GA has to be altered to suit the requirement. There are many facets of operational modification to be introduced on the chromosomes, the operators, and the implementation. The following operations are recommended for appropriate modifications of a GA.

A. Chromosome Representations

Bit string encoding [66] is the most classical approach used by GA researchers because of its simplicity and traceability. A minor modification is the use of Gray code in the binary coding. Reference [67] investigated the use of GA for optimizing functions of two variables and claimed that a Gray code representation worked slightly better than the normal binary representation.

Recently, the direct manipulation of real-value chromosomes [71], [122] raised some considerable interest. This representation was introduced especially to deal with real parameter problems. The work currently taking place by [71] indicates that the floating point representation would be faster in computation and more consistent from the basis of run-to-run. However, the opinion given by [54] suggests that a real-coded GA would not necessarily yield good results 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 pose difficult and sometimes unnatural answers to some optimization problems. The use of other encoding techniques, such as order-based representation [33] (for bin-patching, graph coloring), embedded lists [89] (for factory scheduling problems), variable element lists [89] (for semiconductor layout), and even LISP S-expressions [77] can thus be explored.

B. Objective and Fitness Value

The objective function of a problem is a main source providing the mechanism for evaluating the status of each chromosome. This is an important link between GA and the system. It takes the 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 rescaled to a fitness value [52], [89].

1) Linear Scaling: The fitness value (f_i) of chromosome i has a linear relationship with the objective value O_i as

$$f_i = a \cdot O_i + b \quad (3)$$

where a, b are chosen to enforce the equality of the average objective value and the scaled average fitness values, and cause maximum scaled fitness to be a specified multiple of the average fitness.

2) Power Law Scaling: The actual fitness value is taken as the kth power of the objective value, O_i

$$f_i = O_i^k \quad (4)$$

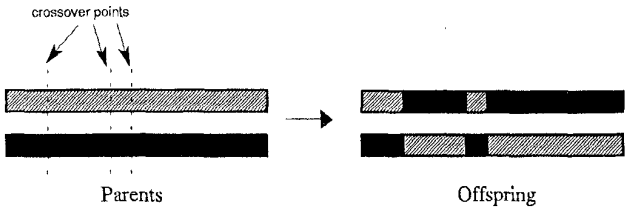


Fig. 7. Example of multipoint crossover, ($m = 3$)

where k is in general problem dependent or even varying during the run.

3) Sigma Truncation: The fitness value f_i of chromosome i is calculated according to

$$f_i = O_i - (\overline{O} - c \cdot \sigma) \tag{5}$$

where c is a small integer, $\overline{O}$ is the mean of the objective values, and σ is the standard deviation in the population. To prevent negative value of f, any negative result $f < 0$ is arbitrarily set to zero.

C. Selection Mechanisms

To generate good offspring, a proficient parent selection mechanism is necessary. This is a process used to determine 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.

Reference [8] presented three measures of performance of selection algorithms, *bias*, *spread*, and *efficiency*. *Bias* defines the absolute difference between actual and expected selection probabilities of individuals. *Spread* is the range in the possible number of trials that an individual may achieved. *Efficiency* is related to the overall time complexity of the algorithms.

Roulette wheel selection tends to give zero bias but potentially inclines to spread unlimitedly. It can generally be implemented with time complexity of the order of $N \log N$ where N is the population size. Stochastic universal sampling (SUS) is another single-phase sampling algorithm with minimum spread, zero bias and the time complexity of SUS is in the order of N [8]. There are other methods can be used such as the ranking scheme [7]. This 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 in the avoidance of premature convergence and to speed up the search when the population approaches convergence [117].

D. Crossover Operations

Although one-point crossover was inspired by biological processes, it has one major drawback in that certain combinations of schema cannot be combined in some situations [89]. A multipoint 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. 7 where multiple crossover points are randomly selected.

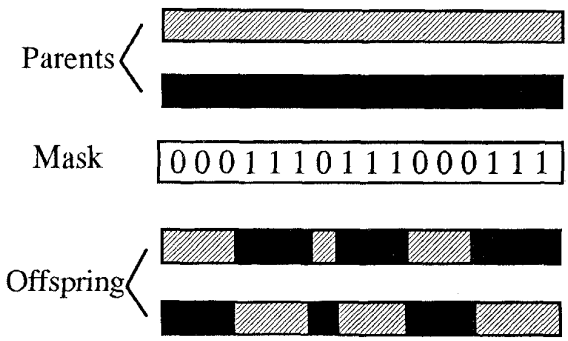


Fig. 8. Example of uniform crossover.

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. 8.

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 which crossover techniques to use is arguable. However, [35] concluded that a two-point crossover seemed to be an optimal number for multipoint crossover. This has since been contradicted by [101] as two-point crossover could perform poorly in a situation where the population has largely converged because of reduced crossover productivity. This low-crossover productivity problem can be resolved by the proposal of reduce-surrogate crossover [12].

Since the uniform crossover exchanges bits rather than segments, it can combine features regardless of their relative locations. This ability may outweigh the disadvantage of destroying building blocks and make uniform crossover a superior operator for some problems [104]. Reference [39] reports on several experiments for various crossover operators. A general comment is that each of these crossovers is particularly useful for some classes of problems and quite poor for others, except that one-point crossover is indicated as a "loser" experimentally.

Some other problem-based crossover techniques have been proposed. Reference [52] described a partially matched crossover (PMX) for the order-based problem. Reference [28] designed an "analogous crossover" for robotic trajectory generation. Therefore, the use of a crossover technique to improve the offspring production, is very much problem oriented. All in all, there is no unified view on this front.

E. Reordering/Inversion

As stated in the building block hypothesis, the order of genes on a chromosome is critical. The purpose of reordering attempts to find gene orders which have better evolutionary potential. Techniques for reordering the positions of genes on the chromosome have been suggested. Reference [52] proposed reversing the order of gene between two randomly chosen positions within the chromosome. Such a technique is known as inversion.

F. Reinsertion

After generating the subpopulation (offspring), there are several representative strategies to replace the old generation that can be proposed.

In the case of generation replacement, the chromosomes in the current population are completely replaced by the offspring [59]. Therefore, the population with size N will generate an equal number of offspring in this strategy. Since the best chromosome of the population may fail to reproduce offspring in the next generation, it is usually combined with an elitist strategy such that one or a number of the best chromosomes can be 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.

G. Probability Rates Setting

The choice of an optimal probability operation rate for crossover and mutation is another controversial debate for both analytical and empirical investigations. The increase of crossover probability causes 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 increase, this would transform the genetic search into a random search, but help to reintroduce the lost genetic material.

As each operator probability may vary through the generations, Davis [30] suggested a linear variation in crossover and mutation probability, with a decreasing crossover rate during the run while the mutation rate increases. Syswerda [105] imposed a fixed schedule for both cases but Booker [12] utilized a dynamically variable crossover rate which is dependent upon the spread of fitness. References [32] and [33] modified the operator probabilities according to the success of generating good offspring. Despite all these suggested methods, the recommendation made by [36] and [59] is the yardstick to apply.

H. Techniques for Changing Environments

There are many instances where GA may be used to optimize the time-variant system characteristics. Should it be adaptive to dynamic signal behavior and/or should it be able to sustain environmental disturbance? Whichever it is, GA has to cope with the requirement such that the time-dependent optima are reached. This is not easy for a standard GA to handle. To ensure that GA responds properly for a changing environment, two basic strategies have been developed.

The first strategy is to expand the memory of the GA in order to build up a repertoire of ready responses for environmental conditions. A typical example is triallelic representation [51]. Triallelic representation consists of a diploid chromosome and a third allelic structure for deciding dominance. The recessive genes provide additional information for the changing environment.

Random immigrants mechanism [62], Triggered hypermutation mechanism [22], [23], and statistical process control [108] are grouped as another type of strategy. This approach increases diversity in the population to compensate for changes encountered in the environment. The random immigrants mechanism replaces a fraction of the GA's population by randomly generated new individuals. It works well in environments where there are occasional, large changes in the location of the optimum. The triggered hypermutation mechanism is an adaptive, mutation-based mechanism to adopt the environmental change. This mechanism temporarily increases the mutation rate to a high value whenever the best time-average performance of the population deteriorates. Statistical process control would then be devised to monitor the best performance of the population such that the GA-based optimization system adapts to the continuous, time-dependent nonstationary environment.

I. Parallel GA

One of the major criticisms of using GA must be the time spent in computation. Sometimes, this can be painfully long. This is understandable as GA is not a mathematically 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 before it can be resolved.

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

There are a number of GA-based parallel methods capable of enhancing the computational speed [15], [18]. The methods of parallelization can be classified as global, migration, and diffusion. 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.

Global GA treats the entire population as a single breeding mechanism. It can be implemented on a shared memory multiprocessor or distributed memory computer. On a shared memory multiprocessor, chromosomes are stored in the shared memory. Each processor evaluates the particular assigned chromosomes. On a distributed memory computer, it is based on the master–slave relationship shown in Fig. 9. One disadvantage of this method is that the slave sits idly by while the master is handling its job. A successful application of this global GA approach can be found in [29] and [37].

Migration GA (coarse grained parallel GA) divides the population into a number of subpopulations, each of which is treated as a separate breeding unit under the control of a conventional GA. To encourage the proliferation of good genetic material throughout the whole population, individual

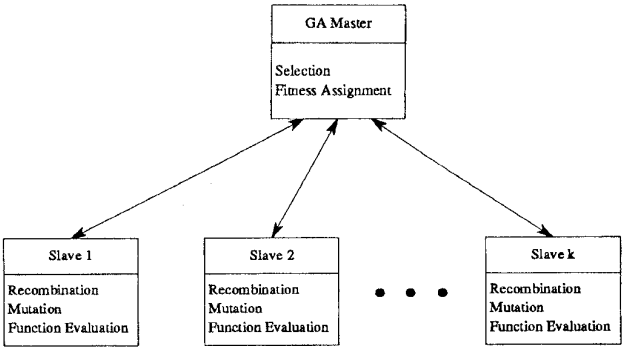


Fig. 9. Global GA.

TABLE II
PSEUDO CODE OF MIGRATION GA

```
-Each node (GA_i)
WHILE not finish
  SEQ
    ... Selection
    ... Reproduction
    ... Reproduction
    ... Evaluation
  PAR
    ... Send emigrants
    ... Receive immigrants
```

migration between the subpopulations occurs from time to time. A pseudo code is expressed in Table II.

Figs. 10–12 show three different topologies in migration. Fig. 10 shows the ring migration topology, where individuals are transferred between directionally adjacent subpopulations. A similar strategy, known as neighborhood migration, is shown on Fig. 11, where migration can be made between nearest neighbors bidirectionally. Unrestricted migration topology is depicted on Fig. 12. Here, individuals may migrate from any subpopulation to another. The individual migrants are then determined according to the appropriate selection strategy.

The topology model of the migration GA is well suited to parallel implementation on multiple instruction multiple data (MIMD) machines. The architecture of hypercubes [24], [112] and rings [56] are commonly used for this purpose. Given the range of possible population topologies and migration paths between them, efficient communications networks should thus be possible on most parallel architecture. This applies to small multiprocessor pathforms or even the clustering of networked workstations.

Diffusion GA (fine grained parallel GA), as indicated in Fig. 13, considers the population as a single continuous structure. Usually, for a connection of massively parallel computers, each individual is assigned to a processing unit which is placed on a 2-D grid topology. Some different topologies have also been studied in this area [3], [9].

The individuals are allowed to breed with individuals contained in a small local neighborhood. This neighborhood is usually chosen from immediately adjacent individuals on the population surface and is motivated by the practical

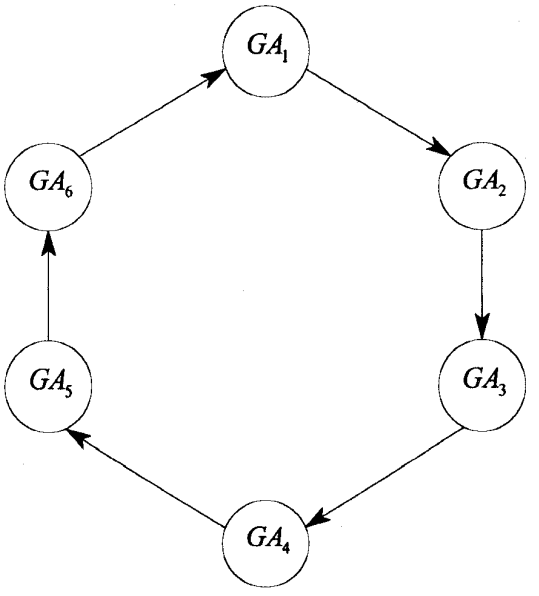


Fig. 10. Ring migration topology.

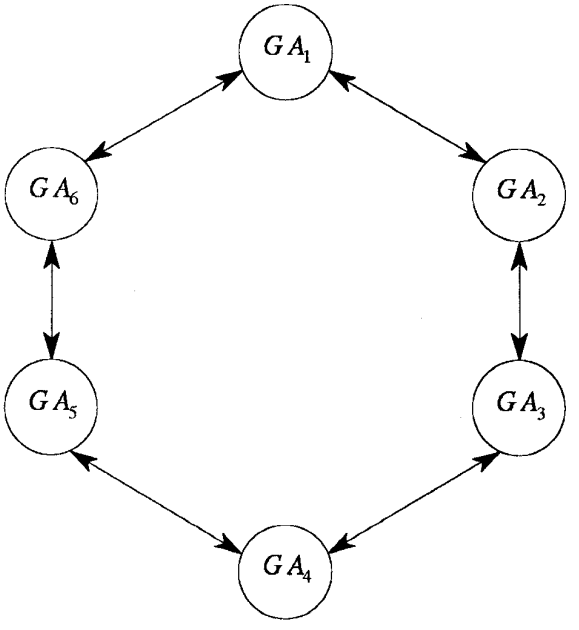


Fig. 11. Neighborhood migration topology.

communication restrictions of parallel computers. The pseudo code is listed on Table III.

Reference [85] introduces a massive parallel GA architecture with population distributed with a 2-D mesh topology. Selection and mating are only possible with neighboring individuals. In addition, [57] and [91] introduce an asynchronous parallel GA, ASPARAGOS system. In this configuration, the GA was implemented on a connected ladder network using transputers with one individual per processor. The practical application of diffusion GA to solve a 2-D bin packing problem has been reported in [78], and the same technique has been implemented to tackle a job shop scheduling problem [106].

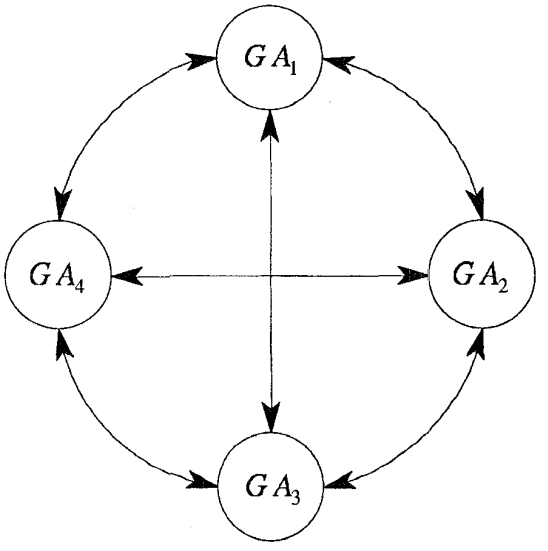


Fig. 12. Unrestricted migration topology.

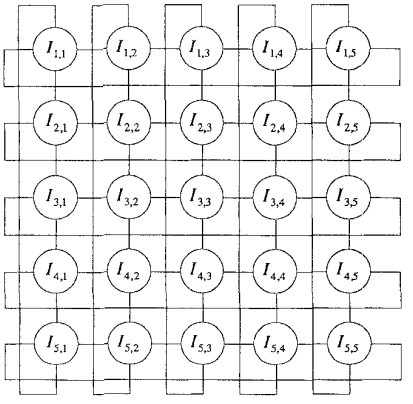


Fig. 13. Diffusion GA.

TABLE III
PSEUDO CODE OF DIFFUSION GA

```
–Each node (I_i,j)
Initialize
WHILE not finished
  SEQ
    ... Evaluate
    PAR
      ... Send self to neighbours
      ... Receive neighbours
    ... Select mate
    ... Reproduce
```

J. Multiobjective GA

Sometimes in everyday life, we seldom experience the luxury of optimizing a single and perfect solution from an objective function. One of the nicer niches of using GA is its capability of solving multiobjective problems [44]. This is particularly useful for meeting system design specifications that are complex, conflicting, and sometimes mathematically difficult.

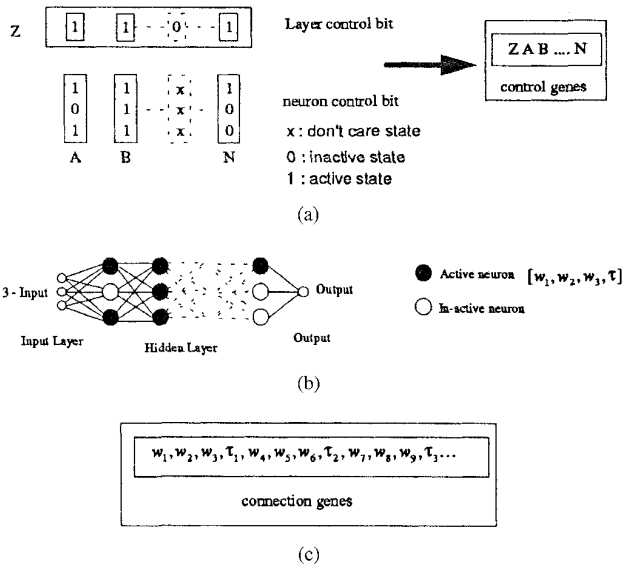


Fig. 14. SGTNN structure. (a) Control genes. (b) Topology of the network by control genes. (c) Associated weighting connection genes.

The solution set of a multiobjective optimization problem consists of all those vectors such that their components cannot all be simultaneously improved. This is known as the concept of Pareto-optimality, and the solution set is known as the Pareto-optimal set [42]. Pareto-optimal solutions are also called nondominated, or noninferior, solutions. Sometimes, a Pareto-optimal set of solutions requires a trade-off with different objective performances. Nevertheless, GA has demonstrated its power in obtaining the Pareto-optimal set instead of a single solution [42], [44], [69].

Of course, a simply way to handle this multiobjective optimization problem is to combine and aggregate the objectives. Several applications using this approach have been reported. The most noted examples of applications are the simple weighted sum approach [70] and the target vector optimization [120].

K. Structured GA

The structured genetic algorithm (SGA) was first proposed by [26]. The chromosome is formulated in a hierarchical structure. Higher-level nodes in the structure govern the activation of the lower-level genes. The deactivated genes provide additional information that allow it to react to a changing environment [27].

SGA has been further developed as a structure optimizer. It has been used for the optimization of structure, like the number of membership sets or rules in fuzzy logic, order of the transfer function [110], and the topology in neural networks (NN) [109]. Fig. 14 illustrates how SGA can be used for NN topology optimization.

The multiple objective approach can be incorporated into SGA (MOSGA) which gives the flexibility of structure optimization. To maintain population diversity and enable searching between different structures, the population is formulated by a number of subgroups [109]. Each subgroup

stores up chromosomes that have a particular class of structure.

V. Advantages of GA: Why Is GA Used?

After discussing the modification of the GA, some insight into why GA has become more and more popular should be given. GA is attractive for a number of reasons.

- It can handle problem constraints by simply embedding them into this chromosome coding.
- Multiobjective problem can be addressed by means of the approach stated in Section IV.
- Since it is a technique independent of the error surface, it is ready to solve multimodal, nondifferientable, noncontinuous, or even NP-complete problems [47].
- Structured GA provides a tool to optimize the topology or the structure in parallel with the parameters of the solution for a particular problem.
- It is a very easy-to-understand technique with very few (or even none) mathematics.
- It can be easily interfaced to existing simulations and models.

VI. Shortcomings of Genetic Algorithms: Problems and Difficulty

Of course, there are some things that GA just cannot or finds difficult to do. The following three phenomena are often encountered.

A. Deception

Some objective functions may be very difficult to optimize by GA. Such functions are referred as GA-deceptive functions [34], [50]. Depending on the nature of the objective functions, very bad chromosomes can be generated by combining good building blocks. This causes the failure of the building block hypothesis.

To illustrate this point in a mathematical fashion, the simplest GA-deceptive function is the minimal deceptive problem [52]. For example, assume that (1, 1) represents the optimal solution for a two-bit objective function, f, and let

$$f(1, 1) > f(0, 0) > f(0, 1) > f(1, 0).$$

The schemata (0, #) does not contain the optimal string (1, 1) as an instance and that leads the GA away from the solution (1, 1). This is only a minimally deceptive problem that is considered to be a partially deceptive function. In a fuller deceptive problem, all low-order schemata containing a suboptimal solution are better than other competing schemata [34].

Three approaches were proposed to deal with deception [89]. The first one is to modify the coding in an appropriated way. This method assumed the prior knowledge of the objective function, which is not always the case. The second approach uses the inversion operation. The last one is the use of messy genetic algorithm [53], [55].

B. Genetic Drift (Bias)

There is no guarantee of obtaining the global optimal point by using GA although it has the tendency to do so. This possibility is reduced if there is a loss of population diversity. As GA seeks out a suboptimal point, the population may converge toward this point and premature convergence occurs. In this case, the global optimal solution can only be obtained by the exploration of mutation in the genetic operations. Such a phenomenon is known as genetic drift [12] and can easily occur when the GA is set with a small population size. Furthermore, this problem is also confronted with when GA is used for solving multiobjective problems for which a single optimal solution is sought instead of a set of Pareto solutions.

A number of techniques have been proposed to limit the effect of genetic drift and maintain population diversity. These include preselection [82], crowding [35], [52], and fitness sharing [45].

C. Real-Time and On-Line Issues

Similar to the other artificial intelligence (AI) techniques and heuristics, the GA is not suited for analyses that would provide guaranteed response times [39], [99]. Moreover, the variance of the response time for an GA system is much larger than the conventional methods. This unfavorable nature limits the use of GA in the real-time problem.

Another characteristic that limits GA's application in on-line application for control is its randomness. In the evolution cycle, it has been noticed that a fitness value is evaluated for each offspring. Since GA has the property of exploring the searching domain, the performance is not guaranteed for one particular offspring. The population may become better and better but the same cannot be said about an individual. Therefore, it is unwise to apply GA directly to a real system without any simulation model.

VII. Applications

Thus far, the pros and cons of using GA have been addressed. Now, we can proceed with our aim to draw your attention by bringing out some practical examples for which GA has been successfully implemented in the area of industrial electronics. This is by no means an exhaustive list, but merely an indicator to realize the trend and potential growth in the future.

A. Parameter and System Identification

In the systems-design phase for both control engineering and signal processing, an adequate but efficient mathematical model is desirable. To obtain such a model, the research work in the area of parameter and system identification has been very active for the last 30 years. The required model, generally in the form of a transfer function or infinite impulse response (IIR) filter, is dominantly addressed. Because of the multimodal error surface of IIR together with its system stability criterion are to be met simultaneously, the gradient type of optimization algorithms would have the problem to obtain an adequate solution. Whereas the use of GA is found

to be an ideal solution for this problem, as GA has the ability to solve multimodal problem with relative ease [93], [121].

The IIR filter in the form of cascade, parallel, or lattice form can be assigned for optimization using GA. This is because of the flexibility of structuring of filter's parameter into the form of a chromosome that can be arranged in a prefixed manner. Therefore, for a stable IIR filter in lattice form, the searching domain of parameters should be set within the range of $+1$ to -1 [116]. For the case of parallel or cascade of second-order form, the coefficients must lie inside the stability triangle [100] for stability assurance.

One further advantage of using GA for identification is its capability to find an optimal order of the IIR filter [110]. This approach of system identification is considered to be a difficult problem for the gradient type of optimization schemes. Here, GA can be applied simultaneously for optimizing the order and coefficient without much effort to alter the chromosome structure.

The use of GA for the purpose of identification is not limited to the IIR design; the power of GA also contributes to nonlinear system identification problems [16]. The GA used for determining the parameters of the Chua's chaotic oscillator [20] proved to be a sound tactic as the temporal series of the state variables and parameters of another Chua's oscillators are inherently known. Synchronization between these two chaotic systems is achieved and is desired for the use of secure communication systems [21], [76].

Reference [43] applied GA for term selection in a nonlinear autoregressive moving average model with exogenous inputs models. An N-array but order-independent chromosome representation is used for the generation of a nonlinear model with a fixed number of terms. Each chromosome consists of a number of integers while each represents a particular term.

B. Control

For its use in control systems engineering, GA can be applied to a number of control methodologies for the improvement of the overall system performance. In most of the controller designs, some parameters are required to be optimized in order to give a better overall control performance. Furthermore, the configuration or the order of the controller may be optimized to reduce the system complexity.

In classical proportional integral derivative (PID) control problems, the required three-terms-parameters which are known as proportional, integral, and derivative gain, should be optimally determined. Despite the method of Ziegler–Nichols (ZN) ultimate-cycle tuning scheme, these parameters can be optimally obtained. Successful implementations are found in the pH neutralization process [115], and the heat exchanger system [72].

In the case of robust control, the principal of the H^{∞} loop shaping design procedure (LSDP) [88] is used to design a precompensator (W_1) and a postcompensator (W_2) to shape the nominal plant (G) so that the singular values of the shaped plant ($G_s = W_2GW_1$) have 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 controller so that an explicit close-loop performance is met.

The advantage of GA in robust controller design are twofold. First, the configurations of the shaping function W_1, W_2 do not necessarily have to be predefined. The optimization of both order and parameter of the shaping function can be undergone simultaneously [48]. This offers the flexibility to search for the parameters of controllers of different degrees of complexity and at the same time it provides a means of achieving the certain optimality. Secondly, the multiple objective approach can be also adopted to further enhance the degree of success of meeting the design criteria of extreme plants [110].

A number of successful applications have been reported in this area. Noticeable contributions are found in magnetically levitated (maglev) vehicles [25], distillation column [110], and the benchmark problem [58].

It has also been reported that GA is being used for designing a sliding mode control system [90]. This is another approach to address the problem of obtaining a static gain matrix in reaching phase which was vividly described in [64].

C. Robotics

Thus far, to the best of the authors' knowledge, the application of GA in robotics has only found a use in robot navigating systems. Considering that such navigation is the art of directing a course for a mobile robot to traverse in a restricted environment, this navigation scheme should then be designed to cope with such constraints, so that the robot is capable of reaching its desired destination without getting lost or crashing into any objects.

Reference [28] demonstrated the application of GA in the trajectory of a three-arm-configuration robot. The end effector's path is of interest. Final trajectory would consist of a series of ordered arm-configurations. Since the number of arm-configurations used is unknown until the solution is found, the coding of GA must be modified to accommodate variable-length and order-dependent strings. Equation (6) shows the string structure of a trajectory consisting of l arm configurations

$$\underbrace{\overbrace{\alpha_{1,1},\ \alpha_{1,2},\ \alpha_{1,n}}^{n\text{-}tuple};\ \alpha_{2,1},\ \cdots,\ \alpha_{l,n}}_{l\ n\text{-}tuples} \tag{6}$$

where each n-$tuple$ represents an arm-configuration.

Michalewicz [89] adopted the order-based coding in his evolutionary navigator (EN). A chromosome in EN is an ordered list of path nodes. Each of the path nodes, apart from the pointer to the next node, consists of x and y coordinates of an intermediate knot point along the path, and a Boolean variable b indicating whether the given node is feasible or not. EN unifies off-line and on-line planning with a simple map of high-fidelity and efficient planning algorithms. The off-line planner searches for the optimal global path from the start to the desired destination, whereas the on-line planner is responsible for handling possible collisions or previously

unknown objects by replacing a part of the original global path by the optimal subtour.

D. Pattern Recognition

The use of GA for pattern recognition has been widely studied. They can be generalized and grouped into two categories.

1) Extraction: GA is applied to generate the image filters for a two-stage target recognition system in such a way that the target image from background clutter is vividly classified from the imaging data [75]. A linear filter is found in the first stage (screener) to select subimages while a filter bank (a set of filters) is being generated in the last stage (classifier) for class decision. Real-number encoding is adopted in this design. Such an approach provides a means to generate suitable filters rapidly with a minimal need for human intervention.

Reference [96] applied GA, based on a minimal subset representation, to perform primitive extraction from geometric sensor data. A novel chromosome representation of a primitive by a minimal subset of member points is proposed. Such a subset is the smallest number of points necessary to define a unique instance of a primitive. For example, three-points is used to define a unique circle; whereas five-points is used for a unique conic. If there are n geometric data points in the input that has an associated index say, from 1–n, then a minimal subset is identified by the indices of its member points. A fixed-sized template is applied to the geometric data to count the total number of points inside the template and then the fitness value is assigned. "Better" would mean a primitive which has more geometric data points within a template.

2) Recognition: A planar object is taken from two viewing positions to be considered as a relationship which is governed by an affine transformation matrix. The transformation parameters can be calculated from a triplet of matched dominant point sets on the boundaries of the images.

However, the factors of different environments, different apparatus, occlusion, noise interference, and distortions would lead to insufficient resolution which, at the end, cause a dramatic change in the number and distributions of the dominant points. Furthermore, certain classes of object shapes, a circle for instance, will result in lacking of prominent dominant points. Reference [114] suggested using standard GA for the transformation parameters calculation to replace the current exhaustive blind searching. The end result is very encouraging and there is room for future development.

A Faceprint system was designed in New Mexico State University [14] for reproducing the feature of a suspected criminal's face. A binary chromosome is used to code the five facial features (mouth, hair, eyes, nose, chin) of a face. Initially, 20 suspects' faces are generated on a computer screen. A witness can then rate each face on a 10-point subjective scale. GA takes that information and follows the evolution cycle to obtain the optimal solution.

E. Speech Recognition

In an automatic speech recognition system, the spoken speech patterns (test pattern) are usually identified with the prestored speech patterns (reference patterns). The comparison of speech signals has a number of difficulties as variations in time and the time scales among them are not fixed. Therefore, time registration of the test and the reference patterns is one of the fundamental problems in the area of automatic isolated word recognition.

The technique of dynamic time warping (DTW) has been developed for this task. The performance is good and accurate. However, the nonlinear time alignment of DTW in which the conflicting issues of optimizing the stringent rule on slope weighting, the nontrivial computation to obtain the K-best paths, and the endpoint constraint relaxation, are not easily solved. The use of GA in this context is the best to apply. Experimental results have shown that the GA approach has a very close recognition performance. In addition, GA has a higher level of confidence in identifying confused input utterances than the conventional approach of DTW, especially for confusable words [79].

F. Engineering Designs

It is interesting to note that GA is not applied only to solve problems that are mathematically oriented in the strictest sense. GA can also be used for engineering designs that include optimization of object shaping, circuit layout and many other applications.

As engineering design is very much an art form, the use of this to create enhanced designs is an object to pursue. GA could thus be used as a tool to aid designers for this purpose. It is anticipated that this method of design is an up-and-coming technology for the creation of futuristic object designs.

A typical but innovated design is an airfoil shapes optimization problem solved by the inverse numerical method [94]. In this task, the associated target pressure distribution is to be acquired by the Navier–Stoke solver. Then, GA is applied to optimize the target pressure distribution so that a supercritical airfoil shape is obtained via design samples from this method.

Another example is the simultaneous optimization of global cells placement and routing in a very large scale integration (VLSI) layout [98]. This is a contrast to the traditional method of layout design for VLSI-chips where the cell and global routes are firstly determined. The GA approach maintains a global view of the layout surface.

On a similar front, [2] introduced GA to find the best state machine assignment for implementing a synchronous sequential circuit, which is considered to be an important step to reducing silicon area or chip count in digital designs. Experiments have indicated that the GA approach yields good results which are often comparable to, or better than, those using established heuristics that embody extensive domain knowledge.

G. Planning and Scheduling

This is another area where GA can be comfortably applied. Optimization is often required for the planning of actions, motions, and tasks. GA has been demonstrated as a power tool for such problems that can be even NP-completed. The applications of GA in the travelling salesman problem [51], pump scheduling in water industry [81], job-shop scheduling [40],

[92], [123], and production–inventory–distribution system [84] are typical examples and have been well recorded.

An interesting maintenance-scheduling problem may also be considered for GA—that is, the activities to be scheduled within a planned maintenance period which requires that they be optimally set. GA is an ideal method for solving constraints problems that are incurred from the complex nature of this kind of situation, where the conflicting issue arises of which parts should be taken out for maintenance at a particular time, especially when time and parts are, in general, interacting against each other. Some initial work has already been given in [80].

H. Classifier System

In a large, complex system, where many control parameters are required to be adjusted in order to keep a system running in an optimal way, the classifier system is generally applied. Rules have been developed to control such complex plants. Such a system involves encoding production rules in string form (classifiers) and learning such rules on-line during their interaction with an arbitrary environment. The application of GA will try to evolve a set of rules to deal with the particular situation. The fitness of a set of rules may be assessed by judging their performance either on the real system itself, or on a computer model of it.

Fogarty [41] used the former method to develop rules for controlling the optimum gas/air mixture in furnaces. Whereas Goldberg modeled a gas-pipeline system to determine a set of rules for pipeline control and gas-leaks detection [52]. Davis and Coomb used a similar approach to design communication network links [31]. There are other more successful applications of GA in this area, like game playing [6] and maze solving, as well as political and economic modeling [46]. There are certain to be many more uses to come as GA is inherently well suited for this type of application.

VIII. Emerging Technology with Genetic Algorithms

GA is considered as one type of emerging technology. But, to integrate GA with other emerging technologies has yet to be mentioned, and understandably, it is worth trying. The reason for this is simple. Thus far, in those emerging technologies, either in neural network (NN) or in fuzzy logic, that we normally come across, their intelligence is merely trying to emulate human brain or behavioral patterns in a mathematically recognizable fashion. Because of its efficiency in modeling for such a complex human system, an optimal model is critical in order to achieve the ultimate goal. At this point, GA may contribute its usefulness in obtaining an optimal model. Therefore, it is only natural to engineers that an integration of GA with either a neural network, or for that matter, with a fuzzy logic system, or both, be used to accomplish a so-called intelligent system. This section attempts to outline some applications of GA in this domain.

A. Integration of Genetic Algorithms and Neural Networks

The use of neural networks (NN) for industrial control has been well accepted. The most noticeable applications are in the areas of telecommunication, active noise control, pattern recognition, prediction and financial analysis, process control, speech recognition, etc. [5], [119]. Combining GA and NN [97] can be generally divided into two broad categories in supportive and collaborative integration.

In supportive integration, GA can assist neural networks in

- selecting features or transforming the feature space used by a neural net classifier [13], [17];
- selecting the learning rule or the parameters that control learning in a neural net; and
- analyzing a neural net [103].

In collaborative integration, GA can be used to optimize NN on the weight parameters and/or topology.

Using GA as a replacement for back propagation for weight optimization does not seem to be very competitive, especially where computational speed is concerned. However, it may be a promising learning method for the reinforcement of training a fully recurrent NN, or even for training networks that are with nondifferentiable transfer neurons.

The contribution of GA's in optimizing a global NN topology optimization is also feasible. Here, GA is used to evolve the network topology and use other local learning methods or GA to fine tune the weights. The applications of GA on neural networks can be shown by the optimization of recurrent neural networks [4] and also feedforward networks [89], [109] where both topology and weight coefficients are globally optimized.

B. Integration of Genetic Algorithms and Fuzzy Logic

In principle, there is no general rule or method to construct the fuzzy rules and to define the membership functions. As a fuzzy concept, the rules and membership functions are defined by the skilled operators and these, obviously, may not be optimal. GA working as the global optimizer is hence applied [73].

References [95] and [107] apply GA to optimize the fuzzy membership functions while [68] uses GA to optimize to optimize both fuzzy membership functions and fuzzy rules simultaneously. A fuzzy controller optimized by GA has been designed on pH control [74] and water pump system [107].

IX. 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 practical applications, is given in this paper. The theme is oriented from an industrial application basis. The paper's purpose is to introduce this emerging technology to engineers who may have little or no knowledge of GA. It is also reasonable to believe that this article contains sufficient, useful material to encourage and awaken the interest to newcomers, so that GA may be implemented to solve their practical problems.

What remains as a challenge to GA is undoubtedly the real-time and adaptive capability. The real-time issue is not so much hinged on the computing speed, since the parallelism of GA should improve the computational speed quite considerably and comfortably. Rather, it is the unpredictability of

GA that is the main cause of concern. As GA is a stochastic, discrete, and nonlinear process, guaranteed response times are not applicable.

Thus far, applying GA in a time-varying system is still in its infancy. Due to the factor of population convergence, GA has difficulty in reacting promptly to the changing environment. Research work in this direction should be encouraged. One possible approach to this problem is to monitor the GA process via some intelligent supervisory scheme.

The integration of GA with other emerging technology such as neural networks and fuzzy logic systems could be another challenging area. The combination of these emerging technologies may not only involve applying GA as a helper to these two, but could result in the emerging technologies being able to assist GA applications. Different combinations may offer us a fruitful result in intelligent system design.

All in all, the knowledge generated from GA over the last 20 or so years has now become mature. The prospect of applying GA for practical applications is good. A considerable growth in the application GA, particularly in the field of industrial electronics, is anticipated for the future.

APPENDIX
GENETIC OPTIMIZATION TOOLS

The study and evaluation of GA are essentially nonanalytic, and largely depend on simulation. While they are strongly application independent, GA software packages have potentially a very broad spectrum of applications. Part of the common software package is briefly introduced and more information can be found in [65].

A. Genetic Algorithm Toolbox in MATLAB

A GA Toolbox was developed [19] for MATLAB [87]. This is GA software which is easy to use, practical, and efficient. It provides a platform for modeling, design, and simulation with an interactive environment and the associated graphical facility.

B. GENESIS

The GENEtic Search Implementation System (GENESIS) was developed by John Grefenstette [61]. It is a function optimization system based on genetic search techniques. As the first widely available GA program, GENESIS has been very influential in stimulating the use of GA, and several other GA packages are generated because of its capability.

C. GENOCOP

GEnetic Algorithm for Numerical Optimization for COnstrained Problems (GENOCOP) is developed by Zbigniew Michalewicz and details can be obtained in [89]. The GENOCOP system was designed to find the global optimum of a function with additional linear equalities and inequalities constraints. Unix and DOS versions are available.

D. GENEsYs

GENEsYs [113] is a GENESIS-based GA implementation which includes extensions and new features for experimental purposes. Different selection schemes like linear ranking, Boltzmann, (μ, λ)-selection, and general extinctive selection variants are included. Crossover operators and self-adaptation of mutation rates are also possible. There are additional data-monitoring facilities, such as recording average, variance, and skew of object variables and mutation rates, and creating bitmap-dumps of the population.

E. TOLKIEN

TOLKIEN (TOoLKIt for gENetics-based applications) version 1.1 [111] is a C++ class library named in memory of J. R. R. Tolkien. A collection of reusable objects has been developed for genetics-based applications. For portability, no compiler-specific or class library-specific features are used. The current version has been compiled successfully using Borland C++ Version 3.1 and GNU C++. TOLKIEN contains a number of useful extensions to the generic GA. For example:

- chromosomes of user-definable types; binary, character, integer, and floating point chromosomes are provided;
- gray code encoding and decoding;
- multipoint and uniform crossover;
- diploidy;
- various selection schemes such as tournament selection and linear ranking; and
- linear fitness scaling and sigma truncation.

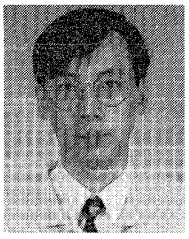
REFERENCES

[1] J. T. Alander, "An indexed bibliography of genetic algorithms: Years 1957–1993," Dept. Informat. Technol., Production Econ., University of Vaasa, Finland, Rep. Ser. 94-1, Feb. 1994.
[2] J. N. Amaral, K. Tumer, and J. Ghosh, "Designing genetic algorithms for the state assignment problem," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 4, Apr. 1995.
[3] E. J. Anderson and M. C. Ferris, "A genetic algorithm for the assembly line balancing problem," Comput. Sci. Dept., Univ. Wisconsin-Madison, Tech. Rep. TR 926, 1990.
[4] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 54–65, Jan. 1994.
[5] K. Asakawa and H. Takagi, "Neural networks in Japan," *Communicat. ACM*, vol. 37, no. 3, pp. 106–112, Mar. 1994.
[6] R. Axelrod, "The evolution of strategies in the iterated prisoner's dilemma," *Genetic Algorithms and Stimulated Annealing*, L. Davis, Ed. New York: Pitman, 1987, pp. 32–41.
[7] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*. J. J. Grefenstette, Ed., Lawrence Erlbaum Associates, 1985, pp. 101–111.
[8] ——, "Reducing bias and inefficiency in the selection algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed., Lawrence Erlbaum Associates, 1987, pp. 14–21.
[9] S. Baluja, "Structure and performance of fine-grain parallelism in genetic search," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993.
[10] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1—Fundamentals," *Univ. Comput.*, vol. 15, no. 2, pp. 58–69, 1993.
[11] ——, "An overview of genetic algorithms: Part 2—Research topics," *Univ. Comput.*, vol. 15, no. 4, pp. 170–181, 1993.
[12] L. Booker, "Improving search in genetic algorithms," in *Genetic Algorithms and Stimulated Annealing*, L. Davis, Ed. New York: Pitman, 1987, pp. 61–73.
[13] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Trans. Neural Networks*, vol. 3, pp. 324–328, 1992.

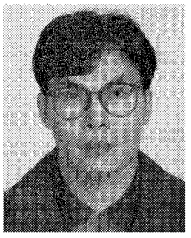
[14] C. Caldwell and V. S. Johnston, "Tracking a criminal suspect through face-space with a genetic algorithm," in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991, pp. 416–421.
[15] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," Illinois Genetic Algorithms Lab., Univ. Illinois at Urbana-Champaign, IlliGAL Rep. 95007, July 1995.
[16] R. C. Caponetto, L. Fortuna, G. Manganaro, and M. G. Xibilia, "Chaotic system identification via genetic algorithm," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 170–174.
[17] E. J. Chang and R. P. Lippmann, "Using genetic algorithms to improve pattern classification performance," *Advances in Neural Information Processing 3,* pp. 797–803, 1991.
[18] A. J. Chipperfield and P. J. Fleming, "Parallel genetic algorithms: A survey," Univ. Sheffield, ACSE Res. Rep. 518, May 1994.
[19] A. J. Chipperfield, P. J. Fleming, and H. Pohlheim, "A genetic algorithm toolbox for MATLAB," in *Proc. Int. Conf. Systems Engineering,* Coventry, U.K., Sept. 6–8, 1994.
[20] L. O. Chua, "The genesis of Chua's circuit," *Int. J. Electron., Communicat.,* vol. 46, no. 4, 1992.
[21] L. O. Chua, K. Eckert, Lj. Kocarev, and M. Itoh, "Experimental chaos synchronization in Chua's circuit," *Int. J. Bifurcation, Chaos,* vol. 2, no. 3, pp. 705–708, 1993.
[22] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," NRL Memo. Rep. 6760, 1990.
[23] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proc. 5th Int. Conf. Genetic Algorithms,* 1993, pp. 523–530.
[24] J. P. Cohoon, W. N. Martin, and D. S. Richards, "A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes," in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991, pp. 244–248.
[25] N. V. Dakev and A. J. Chipperfield, "H_∞ design of an EMS control system for a magle vehicle using evolutionary algorithms," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 141–145.
[26] D. Dasgupta and D. R. McGregor, "A structured genetic algorithm: The model and first results," Univ. Strathclyde, U.K., Res. Rep. IKBS-2-91, Sept. 1991.
[27] ——, "Nonstationary function optimization using the structured genetic algorithm," *Parallel Problem Solving from Nature, 2.* Amsterdam: North Holland, 1992, pp. 145–154.
[28] Y. Davidor, "A genetic algorithm applied to robot trajectory generation," in *Handbook of Genetic Algorithms,* L. Davis, Ed. 1991, pp. 144–165.
[29] R. Davies and T. Clarke, "Parallel implementation of a genetic algorithm," *Contr. Eng. Practice,* vol. 3, no. 1, pp. 11–19, 1995.
[30] L. Davis, "Job shop scheduling with genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms,* J. J. Grefenstette, Ed., 1985, pp. 136–140.
[31] L. Davis and S. Coombs, "Genetic algorithms and communication link speed design: Theoretical considerations," in *Proc. 2nd Conf. Genetic Algorithms,* J. J. Grefenstette, Ed., 1987, pp. 252–256.
[32] L. Davis, "Adapting operator probabilities in genetic algorithms," in *3rd Int. Conf. Genetic Algorithms,* 1989, pp. 61–69.
[33] ——, *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.
[34] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," Tech. Rep. IlliGal 91009, Illigal, Dec. 1991.
[35] K. DeJong, "The analysis and behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1975.
[36] K. A. DeJong and W. M. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Proc. First Workshop Parallel Problem Solving from Nature.* Berlin: Springer Verlag, 1990, pp. 38–47.
[37] N. Dodd, D. Macfarlane, and C. Marland, "Optimization of artificial neural network structure using genetic techniques implemented on multiple transputers," in *Transputing '91,* vol. 2. Amsterdam: IOS Press, 1991, pp. 687–700.
[38] E. H. Durfee, "A cooperative approach to planning for real-time control" in *Proc. Workshop on Innovative Approaches to Planning, Scheduling, and Control,* Nov. 1990, pp. 277–283.
[39] 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.
[40] H.-L. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling, & open-shop scheduling problems," in *Proc. 5th Int. Conf. Genetic Algorithms,* 1993, pp. 375–382.
[41] T. C. Fogarty, "Rule-based optimization of combustion in multiple burner furnaces and boiler plants," *Eng. Applicat. Artificial Intell.,* vol. 1, no. 3, pp. 203–209, 1988.
[42] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjecitve optimization: Formulation, discussion, and generalization," in *Genetic Algorithms: Proc. Fifth Int. Conf.,* S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 416–423.
[43] C. M. Fonseca, E. M. Mendes, P. J. Fleming, and S. A. Billings, "Nonlinear model term selection with genetic algorithms," in *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, Nov. 1993, pp. 27/1–27/8.
[44] C. M. Fonseca and P. J. Fleming, "An overview of evol.utionary Algorithms in multiobjective optimization," Dept. of Automatic Control and Systems Eng., University of Sheffield, U.K., Res. Rep. 527, 1994.
[45] ——, "Multiobjecitve genetic algorithms made easy: Selection, sharing, and mating restriction," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 45–52.
[46] S. Forrest and G. Mayer-Kress, "Genetic algorithms, nonlinear dynamical systems, and models of international security," in *Handbook of Genetic Algorithms,* L. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 166–185.
[47] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: Freeman 1979.
[48] S. J. Goh, D. W. Gu, and K. F. Man, "Multi-layer genetic algorithms in robust control system design," *Control '96,* U.K., Sept. 1996.
[49] D. E. Goldberg, "Alleles, locis, and the TSP," in *Proc. 1st Int. Conf. Genetic Algorithms,* 1985, pp. 154–159.
[50] ——, "Simple genetic algorithms and the minimal deceptive problem," in *Genetic Algorithms and Stimulated Annealing,* L. Davis, Ed. New York: Pitman, 1987, pp. 74–88.
[51] D. E. Goldberg and R. E. Smith, "Nonstationary function optimization using genetic dominance and diploidy," in *Proc. 2nd Int. Conf. Genetic Algorithms,* 1987, pp. 59–68.
[52] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.
[53] ——, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems,* vol. 3, pp. 495–530, 1989.
[54] ——, "Real-coded genetic algorithms, virtual alphabets, and block," Univ. Illinois, Tech. Rep. 90001, Sept. 1990.
[55] D. E. Goldberg, K. Deb, and B. Korb, "Do not worry, be messy," in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991.
[56] V. Gordon and D. Whitley, "Serial and parallel genetic algorithms as function optimizer," in *Proc. 5th Int. Conf. Genetic Algorithms,* 1993, pp. 177–183.
[57] M. Gorges-Schleuter, "ASPARAGOS an asynchronous parallel genetic optimization strategy," in *Proc. 3rd Int. Conf. Genetic Algorithms,* 1989, pp. 422–427.
[58] S. F. Graebe, "Robust and adaptive control of an unknown plant: A benchmark of new format," in *IFAC World Congr. Proc.,* 1993, vol. 3, pp. 165–168.
[59] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-16, pp. 122–128, Jan./Feb. 1986.
[60] J. J. Grefenstette and J. Baker, "How genetic algorithms work: A critical look at implicit parallelism," in *Proc. 3rd Int. Conf. Genetic Algorithm,* 1989.
[61] J. J. Grefenstette, *A User's Guide to GENESIS v5.0,* Naval Res. Lab., Washington, D.C., 1990.
[62] ——, "Genetic algorithms for changing environments," *Parallel Problem Solving from Nature, 2.* Amsterdam: North Holland, 1992, pp. 137–144.
[63] ——, "Deception considered harmful," in *Foundations of Genetic Algorithms, 2,* L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 75–91.
[64] B. S. Heck, S. V. Yallapragada, and M. K. H. Fan, "Numerical methods to design the reaching phase of output feedback variable structure control," *Automatica,* 1993.
[65] J. Heitkoetter and D. Beasley, Eds. (1994). *The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ).* [Online]. Available: USENET:comp.ai.genetic.
[66] J. H. Holland, *Adaption in Natural and Artificial Systems.* Cambridge, MA: MIT Press, 1975.
[67] R. B. Hollstien, "Artificial genetic adaptation in computer control systems," Ph.D. dissertation, University of Michigan, Ann Arbor, 1971.
[68] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.,* vol. 3, pp. 129–139, May 1995.
[69] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm," Univ. Illinois at Urbana-Champaign, Urbana, IL, IlliGAL Rep. 93005.
[70] W. Jakob, M. Gorges-Schleuter, and C. Blume, "Application of genetic algorithms to task planning and learning," *Parallel Problem Solving from Nature, 2,* pp. 291–300, 1992.

[71] 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,* July 1991, pp. 31–36.
[72] A. H. Jones and P. B. De Moura Oliveira, "Genetic auto-tuning of PID controllers," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 141–145.
[73] C. L. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert,* vol. 6, no. 2, pp. 26–33, 1991.
[74] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.,* vol. 1, pp. 46–53, 1993.
[75] A. J. Katz and P. R. Thrift, "Generating image filters for target recognition by genetic learning," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 16, pp. 906–910, Sept. 1994.
[76] Lj. Kocarev, K. S. Halle, K. Eckert, U. Parlitz, and L. O. Chua, "Experimental demonstration of secure communications via chaos synchronization," *Int. J. Bifurcation, Chaos,* vol. 2, no. 3, pp. 709–713, 1992.
[77] J. Koza, "Evolution and co-evolution of computer programs to control independently-acting agents," in *Animals to Animals,* J. A. Ineger and S. W. Willson, Eds. Cambridge, MA: MIT Press/Bradford Books, 1991.
[78] B. Kröger, P. Schwenderling, and O. Vornberger, "Parallel genetic packing on transputers," in *Parallel Genetic Algorithms: Theory and Applications.* Amsterdam: IOS Press, 1993, pp. 151–185.
[79] S. Kwong, A. C. L. Ng, and K. F. Man, "Improving local search in genetic algorithms for numerical global optimization using modified GRID-point search technique," in *1st IEE/IEEE Int. Conf. on GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 419–423.
[80] W. B. Langdon. (1995). *Scheduling Planned Maintenance of the (U.K.) National Grid.* [Online]. Available: cs.ucl.ac.uk:/genetic/papers/grid_aisb-95.ps.
[81] G. Mackle, D. A. Savic, and G. A. Walters, "Application of genetic algorithms to pump scheduling for water supply," in *1st IEE/IEEE Int. Conf. on GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K,, 1995, pp. 400–405.
[82] S. W. Mahfoud, "Crowding and preselection revisited," Dept. Comput. Sci., Univ. Illinois at Urbana-Champaign, IlliGAL Rep. 92004, Apr. 1992.
[83] ——, "Population sizing for sharing methods," Dept. Comput. Sci., Univ. Illinois at Urbana-Champaign, IlliGAL Rep. 94005, Aug. 1994.
[84] K. L. Mak and Y. S. Wong, "Design of integrated production-inventory-distribution systems using genetic algorithm," in *1st IEE/IEEE Int. Conf. on GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 454–460.
[85] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms,* 1989, pp. 428–433.
[86] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Networks,* vol. 5, pp. 39–53, Jan. 1994.
[87] *MATLAB User's Guide.* The MathWorks, 1991.
[88] D. McFarlane and K. Glover, "Robust controller design using normalized coprime factor plant descriptions," in *Information and Control Sciences.* Berlin: Springer-Verlag, 1990.
[89] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program,* 2nd Ed. Berlin: Springer-Verlag, 1994.
[90] N. H. Moin, A. S. I. Zinober, and P. J. Harley, "Sliding mode control design using genetic algorithms," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 238–244.
[91] H. Mühlenbein, "Parallel genetic algorithms, population genetics, and combinatorial optimization," in *Parallelism, Learning, Evolution.* Berlin: Springer-Verlag, 1989, pp. 398–406.
[92] R. Nakano, "Conventional genetic algorithms for job-shop problems," in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991, pp. 474–479.
[93] R. Nambiar and P. Mars, "Adaptive IIR filtering using natural algorithms," in *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, Nov. 1993, pp. 20/1–20/10.
[94] S. Obayashi, "Genetic algorithm for aerodynamic inverse optimization problems," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K,, 1995, pp. 7–12.
[95] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.,* vol. 24, pp. 39–47, 1994.
[96] G. Roth and M. D. Levine, "Geometric primitive extraction using a genetic algorithm," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 16, pp. 901–905, Sept. 1994.
[97] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proc. COGANN-92 Int. Workshops on Combination of Genetic Algorithms and Neural Networks,* Baltimore, MD, June 6, 1992.
[98] V. Schnecke and O. Vornberger, "Genetic design of VLSI-layouts," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 430–435.
[99] K. G. Shin and P. Ramanathan, "Real-time computing: A new discipline of computer science and engineering," in *Proc. IEEE,* vol. 82, no. 1, Jan. 1994.
[100] J. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Mag.,* pp. 4–21, Apr. 1989.
[101] W. M. Spears and K. DeJong, "An analysis of multi-point crossover," in *Foundations of Genetic Algorithms,* G. J. E. Rawlins, Ed. 1991, pp. 301–315.
[102] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer,* pp. 17–26, June 1994.
[103] K. Suzuki and Y. Kakazu, "An approach to the analysis of the basins of the associative memory model using genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms,* 1991, pp. 539–546.
[104] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms,* 1989, pp. 2–9.
[105] ——, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms,* pp. 332–349, 1991.
[106] H. Tamaki and Y. Nichikawa, "A paralleled genetic algorithm based on a neighborhood model and its application to job shop scheduling," *Parallel Problem Solving from Nature, 2,* pp. 573–582, 1992.
[107] K. S. Tang, K. F. Man, and C. Y. Chan, "Fuzzy control of water pressure using genetic algorithm," in *Proc. IFAC Workshop on Safety, Reliability, and Applications of Emerging Intelligent Control Technologies,* Hong Kong, Dec. 1994, pp. 15–20.
[108] K. S. Tang, K. F. Man, and S. Kwong, "GA approach to time-variant delay estimation," in *Int. Conf. Control, Information,* Hong Kong, June 5–9, 1995.
[109] K. S. Tang, C. Y. Chan, K. F. Man, and S. Kwong, "Genetic structure for nn topology and weights optimization," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 250–255.
[110] K. S. Tang, K. F. Man, and D. W. Gu, "Structured genetic algorithm for robust H_∞ control system design," *IEEE Trans. Ind. Electron.,* this issue, pp. 575–582.
[111] Y. C. Tang, *Tolkien Reference Manual,* Dept. Comput. Sci., Chinese Univ. Hong Kong, 1994.
[112] R. Tanse, "Distributed genetic algorithms," in *Proc. 3rd. Int. Conf. Genetic Algorithms,* 1989, pp. 434–439.
[113] B. Thomas, *Users Guide for GENEsYs,* System Analysis Research Group, Dept. Comput. Sci., Univ. Dortmund, 1992.
[114] P. W. M. Tsang, "A genetic algorithm for affine invariant object shape recognition," in *1st IEE/IEEE Int. Conf. GA's in Engineering Systems: Innovations and Applications,* Sheffield, U.K., 1995, pp. 293–298.
[115] P. Wang and D. P. Kowk, "Optimal design of pid process controllers based on genetic algorithms," *Contr. Eng. Practice,* vol. 2, no. 4, pp. 641–648, 1994.
[116] M. S. White and S. J. Flockton, "A comparative study of natural algorithms for adaptive IIR filtering," in *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, Nov. 1993, pp. 22/1–22/8.
[117] D. Whitley, "The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best," in *Proc. 3rd Int. Conf. Genetic Algorithms,* J. D. Schaffer, Ed., 1989, pp. 116–121.
[118] ——, "A genetic algorithm tutorial," Dept. Comput. Sci, Colorado State Univ., Tech. Rep. CS-93-103, Nov. 1993.
[119] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: Applications in industry, business, and science," *Communicat. ACM,* vol. 37, no. 3, pp. 93–105, Mar. 1994.
[120] 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, pp. 211–225, 1992.
[121] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," in *Workshop on Natural Algorithms in Signal Processing,* Chelmsford, Essex, Nov. 1993, pp. 4/1–4/8.
[122] A. H. Wright, "Genetic algorithms for real parameter optimization," *Foundations of Genetic Algorithms,* J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 205–218.
[123] T. Yamada and R. Nakano, "A genetic algorithm applicable to large-scale job-shop problems," *Parallel Problem Solving From Nature, 2,* pp. 281–290, 1992.

K. F. Man (M'91), for a photograph and biography, see this issue, p. 518.



K. S. Tang received the B.Eng. degree (Hons.) in electrical and electronics engineering from the University of Hong Kong in 1988 and the M.Sc. degree from City University of Hong Kong in 1992. He is currently working toward the Ph.D. degree in the Department of Electronic Engineering, City University of Hong Kong. His research interests include genetic algorithms, active noise control, chaotic and nonlinear system control.



S. Kwong (M'93) received the M.A.Sc. degree in electrical engineering from the University of Waterloo, Canada, in 1985.

He joined Control Data Canada as a diagnostic engineer and participated in the design and development of diagnostic software for the Cyber mainframe computer series. In 1987, he joined Bell Northern Research as a Member of Scientific Staff responsible for the development of the telecommunication systems in both voice and data network. In 1989, he joined City University of Hong Kong as a Lecturer in the Department of Electronic Engineering and later on moved to the Department of Computer Science. He is involved actively in both academic and industrial projects and publishes widely in the areas of speech processing, computer system, pattern recognition, and Chinese computing.