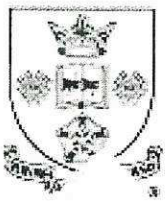
GENETIC ALGORITHMS
IN
CONTROL SYSTEMS ENGINEERING

P. J. FLEMING



R. C. PURSHOUSE

Department of Automatic Control and Systems Engineering
University of Sheffield
Sheffield, S1 3JD
UK

Research Report No. 789
May 2001

Genetic algorithms in control systems engineering

P. J. Fleming and **R. C. Purshouse**
Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK

Abstract

Genetic algorithms (GAs) are global, parallel, stochastic search methods, founded on Darwinian evolutionary principles. Many variations exist, including genetic programming and multiobjective algorithms. During the last decade GAs have been applied in a variety of areas, with varying degrees of success within each. A significant contribution has been made within control systems engineering. GAs exhibit considerable robustness in problem domains that are not conducive to formal, rigorous, classical analysis. They are not limited by typical control problem attributes such as ill-behaved objective functions, the existence of constraints, and variations in the nature of control variables. GA software tools are available, but there is no 'industry standard'. The computational complexity of the GA has proved to be the chief impediment to real-time application of the technique. Hence, the majority of applications that use GAs are, by nature, off-line. GAs have been used to optimise both structure and parameter values for both controllers and plant models. They have also been applied to fault diagnosis, stability analysis, robot path-planning, and combinatorial problems (such as scheduling and bin-packing). Hybrid approaches have proved popular, with GAs being integrated in fuzzy logic and neural computing schemes. The GA has been used as the population-based engine for multiobjective optimisers. Multiple, Pareto-optimal, solutions can be represented simultaneously. In such schemes, a decision-maker can lead the direction of future search. Interesting future developments are anticipated in on-line applications and multiobjective search and decision-making.

1. Introduction

The *genetic algorithm* (GA) has arisen from a desire to model the biological processes of natural selection and population genetics, with the original aim of designing autonomous learning and decision-making systems [Holland, 1975]. Since its introduction, and subsequent popularisation [Goldberg, 1989], the GA has been frequently utilised as an alternative optimisation tool to conventional methods. The correctness of the GA as an abstraction of natural evolution has been challenged, for example by Channon and Damper [2000], but this issue should not be of undue concern to the engineer, who is using the GA for its robust search and optimisation properties.

Several analogous algorithms have been proposed in the literature, such as *evolution strategies* (ES) and *evolutionary programming* (EP). These, together with GAs, have been classified under the umbrella group of *evolutionary algorithms* (EAs) [Spears *et al*, 1993].

This article describes how the genetic algorithm methodology can be applied to problems in control systems engineering. The suitability of the GA towards various types of problem is discussed, and methods for incorporating the characteristics of control problems, such as constraints on actuator performance, are outlined.

The application of GAs to control can broadly be classified into two distinct areas: off-line design and on-line optimisation. Off-line applications have proved to be the most popular and successful. On-line applications tend to be quite rare because of the difficulties associated with using a GA in real-time and directly influencing the operation of the system. GAs have been applied to controller design and to system identification. In each case, either the parameters or the structure can be optimised, or – potentially – both. Other applications include fault diagnosis, stability analysis, sensor-actuator placement, and other combinatorial problems. This article considers examples from the literature for each class of problem.

The article concludes by offering future perspectives on the direction of EA research, with particular attention to issues of concern to the control engineer.

2. What are genetic algorithms?

2.1. Overview

Genetic algorithms (GAs) are global, parallel, search and optimisation methods, founded on Darwinian principles [Darwin, 1859]. They work with a population of potential solutions to a problem. Each individual within the population represents a particular solution to the problem, generally expressed in some form of genetic code. The population is evolved, over generations, to produce better solutions to the problem. A schematic of the algorithm is shown in Figure 1.

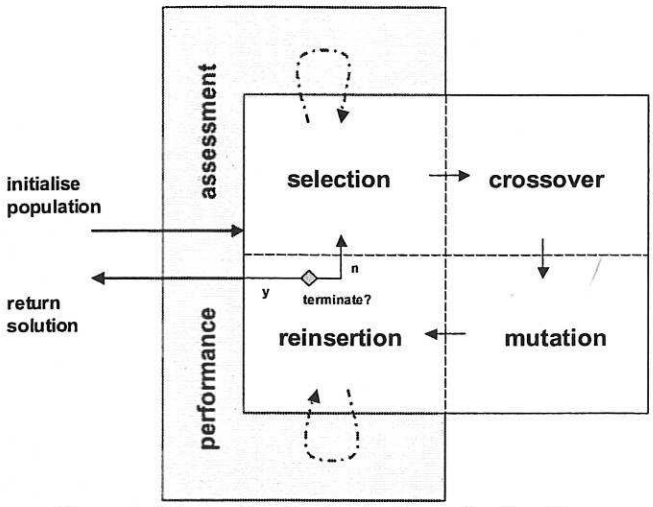


Figure 1: Schematic of a standard genetic algorithm

Each individual within the population is assigned a fitness value, which expresses how good the solution is at solving the problem. The fitness value probabilistically determines how successful the individual will be at propagating its genes (its code) to subsequent generations. Better solutions are assigned higher values of fitness than worse performing solutions.

Evolution is performed using a set of stochastic genetic operators, which manipulate the genetic code. Most genetic algorithms include operators that select individuals for reproduction, produce new individuals based on those selected, and determine the composition of the population at the subsequent generation. *Crossover* and *mutation* are two well-known operators.

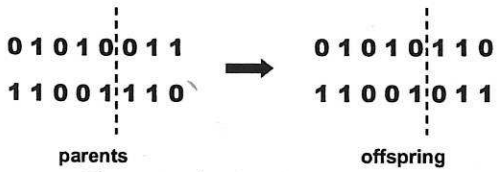


Figure 2: Single point crossover

The crossover operator involves the exchange of genetic material between chromosomes (parents), in order to create new chromosomes (offspring). Various forms of this operator have been developed. The simplest form, single point crossover, is illustrated in Figure 2. This operator selects two parents, chooses a random position in the genetic coding, and exchanges genetic information to the right of this point, thus creating two new offspring.

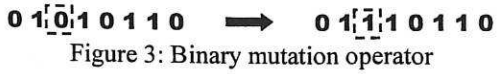


Figure 3: Binary mutation operator

The mutation operator, in its simplest form, makes small, random, changes to a chromosome. For a binary encoding, this involves swapping gene **1** for gene **0** with small probability (typically around one percent) for each bit in the chromosome, as shown in Figure 3.

Once the new generation has been constructed, the processes that result in the subsequent generation of the population are begun once more.

The genetic algorithm explores and exploits the search space to find good solutions to the problem. It is possible for a GA to support several dissimilar, but equally good, solutions to a problem, due to its use of a population.

Despite the simple concepts involved, genetic algorithms can become quite complicated. Many variations have been proposed since the first GA was introduced. Rigorous mathematical analysis of the GA is difficult and is still incomplete.

Genetic algorithms are robust tools, able to cope with discontinuities and noise in the problem landscape. Inclusion of domain-specific heuristics is not a pre-requisite, although it may improve the performance of a GA. They have proved useful at tackling problems that cannot be solved using conventional means.

2.2. Landscapes

The genetic algorithm seeks to maximise the mean fitness of its population, through the iterative application of the genetic operators previously described. The fitness value of a solution in the GA domain corresponds to a *cost* value in the problem domain. An explicit mapping is made between the two domains. 'Cost' is a term commonly associated with traditional optimisation problems. It represents a measure of performance: namely, the lower the cost, the better the performance. Optimisers seek to minimise cost. Hence, it is evident that, by maximising fitness, the GA is effectively minimising cost. Raw performance measures must be translated to a cost value. This process is usually straightforward for single objective problems, but becomes more complicated in the multiobjective case.

Every possible decision vector has an associated cost value and fitness value. The enumeration of all such vectors leads to the *cost landscape* and *fitness landscape* for the problem. For a problem with two decision variables, the cost and fitness landscapes will each be three-dimensional. An example is given in Figure 4. In general, if a problem has n decision variables (is n-dimensional), then the corresponding landscapes will be $n+1$ dimensional. The nature of a cost landscape depends on the chosen mapping from the vector of raw performance measure to the scalar cost value. The nature of the scalar fitness value subsequently depends on the translation from cost to fitness.

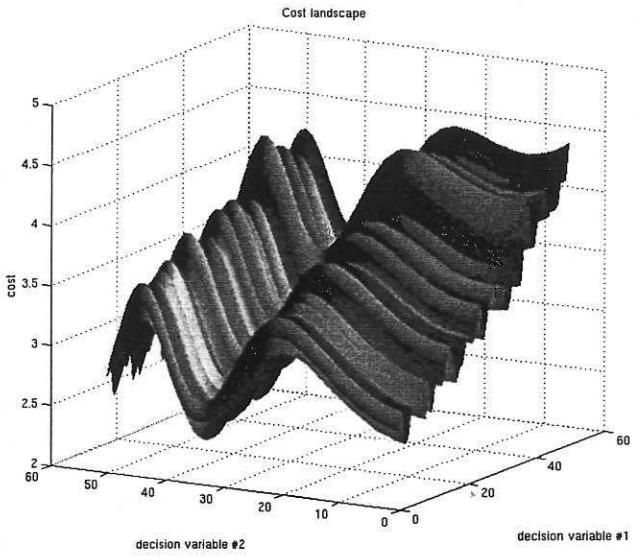


Figure 4: A multimodal cost landscape

2.3. Diversity

Many variations on the standard genetic algorithm, as presented by Goldberg [1989], can be found in the literature. Modifications have been motivated by a desire to improve the performance of the GA, and to adapt it to particular problem domains. It may be more helpful or appropriate to regard evolutionary computing as a general problem-solving methodology, rather than a specific parameter-less tool.

Virtually every aspect of the GA has been exposed to experimentation. As a note of caution, the results of such changes are often inconclusive and are frequently based on limited empirical testing. A very brief summary of key developments is presented below:

- **Population** – The size of the population has been of standard concern to both theorists and implementers. A population of between twenty and one hundred chromosomes is normally sufficient for most applications. The encoding of potential solutions to form chromosomes has also been the subject of intense research. Binary, or its *Gray* variant, encoding is the traditional approach, but direct floating-point representations of design parameters are becoming increasingly popular [Michalewicz, 1996].
- **Fitness assignment** – Techniques for the conversion of the raw performance of a potential solution to a GA fitness value have also received much attention. Fitness is often taken as absolute, prior to normalisation using the population average but, alternatively, ranking techniques may be used. The main aims are to prevent premature convergence (early in the search), and to prevent directionless search late in the search. Ranking is, arguably, the most effective method of achieving this.
- **Selection** – The standard roulette wheel selection method is known to produce biased results, leading to a phenomenon known as *genetic drift*. Two central aims in the development of alternatives are to eliminate statistical bias and to achieve potential parallelism. Other selection methods have been proposed, such as tournaments between two individuals (which achieves good parallelism), and stochastic universal sampling (which is unbiased) [Baker, 1987]. Note that a trade-off has been shown to exist between the two aims cited above.
- **Genetic manipulation** – Genetic operators have been subject to intensive discussion, over both the composition and purpose of the various operators. Some researchers

have abandoned recombination, whilst others regard the effect of mutation as minimal. Essentially, recombination tends to direct the search to superior areas of the search space, whilst mutation acts to explore new areas of the search space and to ensure that genetic material cannot be irretrievably lost. Choice of genetic operators must be made together with choice of representation.

- **Iteration** – GAs evolve a population over a number of generations. The exact number depends on the speed with which convergence can be achieved, and is dependent on the interplay between the GA construction and the type of problem under consideration. The composition of each new generation must be chosen. Typically, this will include offspring produced as a result of genetic operators acting on old individuals, some remnants of the past population, and possibly a few randomly generated individuals (see Figure 5). The exact proportion of each tends to vary from implementation to implementation, but offspring usually dominate the new population. The ratio of offspring to population size is termed the *generational gap*. It may be a static value, or may vary dynamically during the course of a run. *Elitism* is the term given to describe the deliberate introduction of high-achieving (relative to the current population) past individuals into the new population.

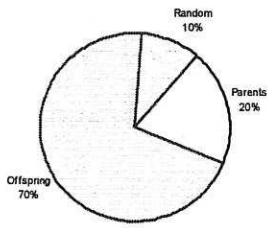


Figure 5: Typical composition of the new generation

- **New operators** – Various new operators have been introduced to address problems discovered in application. For example, *fitness sharing* can be used to encourage *niching* behaviour (sub-population formation at different, comparatively optimal, landscape peaks). *Mating restriction* can be applied in cases where crossover between largely different solutions is unlikely to create good offspring (poor offspring are commonly described as *lethals*).

A particular area of interest is the endeavour to incorporate further parallelism within the GA methodology in order to improve the efficiency of the algorithm. Three main categories of parallel GA (PGA) can be defined, namely *global*, *migration*, and *diffusion* algorithms. Global PGAs treat the entire population as a single breeding unit and aim to exploit the inherent parallelism of the algorithm. Farmer-worker systems are a typical implementation, in which the workers carry out performance evaluations, or conduct genetic operations. In a migration-based PGA, the population is distributed amongst semi-isolated groups. From time to time, migration of individuals within the groups occurs. Diffusion PGAs are based on a local neighbourhood selection mechanism. The population is treated as a single, continuous, structure. Breeding is restricted to adjacent individuals. This type of scheme tends to give rise to clusters of individuals of similar genetic material and fitness, known as 'virtual islands'. Chipperfield and Fleming [1995] provide a broad overview and comparison of parallel GAs.

GAs have also been utilised as a component of hybrid problem-solving tools, including elements such as hill-climbing, simulated annealing, neural networks, Bayesian belief networks, and fuzzy logic.

Two key developments that have arisen from the GA are *genetic programming* (GP) and *multiobjective evolutionary algorithms* (MOEAs). General introductions are provided in the following subsections.

2.3.1. Genetic programming

Genetic programming represents a major variation on the GA. It was developed by Koza [1992], with the original purpose of generating and evaluating entire computer programs. The algorithm fundamentally resembles a GA but application of the operators requires special care. GP evaluates and manipulates variable length structures, in contrast to the generally fixed length chromosomes of a GA. The structures are composed of functions and terminals (potential inputs) that are defined in a library, prior to the execution of the GP. The maximum depth of any structure is, usually, also pre-defined.

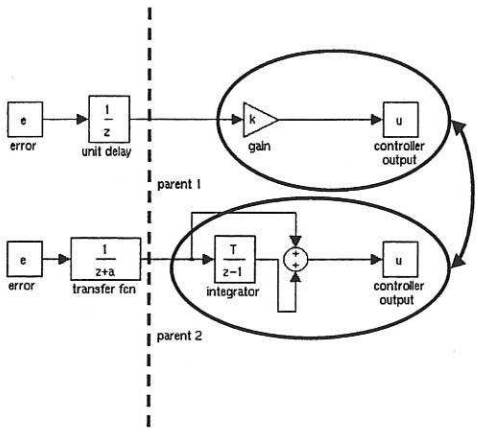


Figure 6: Single point crossover for GP using block diagrams

GP uses a parse tree structure that is very similar to the Lisp programming language. However, the GP approach admits any problem for which the solution can be represented as a structure. Applications have involved manipulation of structures such as neural networks, system block diagrams, circuits, and equations. A single point crossover operator for a block diagram representation is shown in Figure 6. A simple mutation operator is illustrated in Figure 7. This operator swaps, with small probability, a block within a particular solution for one from the library of possible blocks. GP embodies an entire field of research in its own right. Refer to Swain and Zalzala [1998] for a detailed synopsis of GP trends and applications.

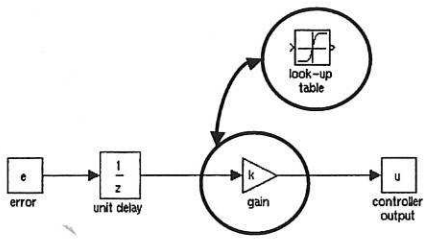


Figure 7: Mutation for GP using block diagrams

2.3.2. Multiobjective evolutionary algorithms

Real-world problems usually involve the simultaneous consideration of multiple performance criteria. These objectives are often non-commensurable and are frequently in conflict with one another. *Trade-offs* exist between some objectives, where advancement in one objective will cause deterioration in another. It is very rare for problems to have a single solution; rather a family of *non-dominated* solutions will exist. These *Pareto-optimal* (PO) solutions are those for which no other solution can be found which improves on a particular objective without detriment to one or more other objectives. The concept of Pareto optimality is illustrated in Figure 8.

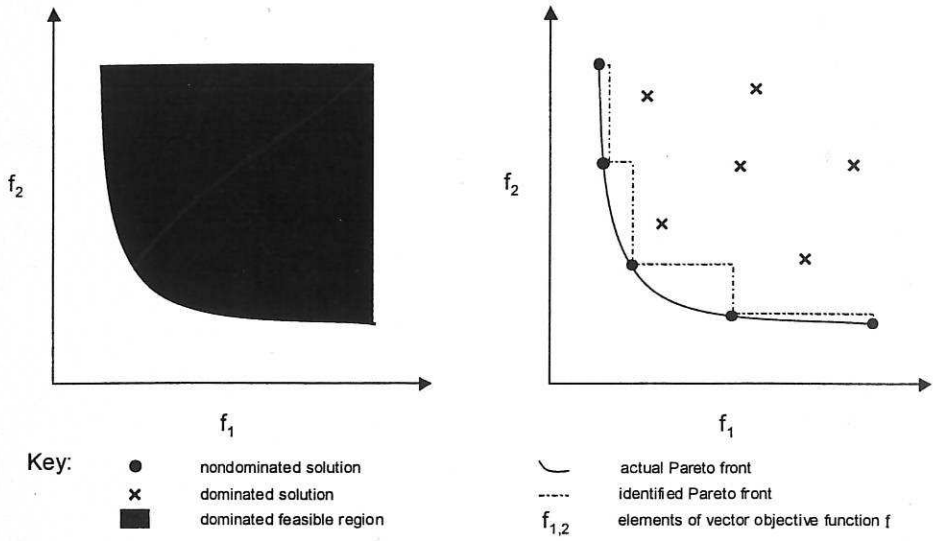


Figure 8: Pareto optimality

Evolutionary algorithms are a suitable technique for multiobjective optimisation. Due to their population-based nature, they are capable of supporting several different solutions simultaneously. The robustness of the GA in the face of ill-behaved problem landscapes increases the value of their utility. Research into multiobjective evolutionary algorithms is still in its infancy, and is likely to prove a highly fruitful line of investigation in the coming years. One of the first approaches to utilise the concept of Pareto optimality was Fonseca and Fleming's [1993] multiobjective genetic algorithm (MOGA), which tends to be the favoured approach for control engineers. Several excellent surveys of multiobjective evolutionary algorithm (MOEA) activity can be found, namely: [Veldhuizen and Lamont, 2000; Coello, 1999; Deb, 1999; Fonseca and Fleming, 1995].

In the past, multiobjective problems have been cast as, effectively, single objective problems by constructing a utility function describing the relative importance of each objective. For example, in linear quadratic regulator design, the competing objectives of error and control size have in the past been combined as a weighted-sum of quadratic measures. The utility function is defined prior to the optimisation procedure. It requires in-depth information concerning the various trade-offs and valuation of each individual. This data is not commonly fully available in practice. In contrast, the GA selection operator can be used to identify degrees of Pareto optimality, thus enabling objectives to be handled individually. Hence, the requirement for a forced combination of objectives and the need for *a priori* information are both avoided. Indeed, this kind of search can help identify the existence and nature of specific trade-offs.

The central theme of MOEA research to date has been the search for a problem's *Pareto-front* (the set of non-dominated solutions). This set can be quite large and, hence, preference

information may be usefully incorporated in order to direct the search to useful parts of the trade-off surface. Incorporation of designer preferences within a MOEA-based tool is a crucial area for further research. Fonseca and Fleming [1998] proposed a scheme for the progressive articulation of preferences by means of aspiration levels (specified in terms of goals and priorities). The authors' *preferability* relation can be considered as a unification of several popular multiobjective decision strategies. Coello [2000] has undertaken a survey of existing approaches to preference handling within MOEAs.

MOEAs can be applied to a wide range of design problems, encompassing many different fields. For example, a MOGA has been applied to the optimisation of radiotherapy treatment planning [Haas *et al*, 1997], in which the objectives are to deliver a high dose to the target area, whilst sparing the organs at risk, and minimising the dose to other healthy tissue. MOEAs have also been applied to engineering design problems such as supersonic wing-shape optimisation [Obayashi *et al*, 2000] and automotive engine design [Fujita *et al*, 1998]. Control-related applications are described in Section 4, many of which extend design capabilities of GA search methods based on single objectives.

3. How can GAs be of benefit to control?

This section outlines the motivation for using GAs in control systems engineering. The attractiveness of GAs for use in complex problems is described, together with challenges that may limit their application. The case for using conventional approaches instead of GAs is also presented. The methods by which standard control problem attributes, such as constraints, can be handled by a GA are discussed. Finally, a summary of available GA tools is presented.

3.1. Suitability

Many exponents of evolutionary algorithms cite the genre's generic nature as a major advantage. GAs can be applied to a wide-range of problems without significant modification. However, the GA is sometimes perceived as a tool that will provide mediocre results in a problem domain when compared with domain-specific methods. This criticism is further explored in the later sections of this article. It should also be noted that GA parameters (population size, mutation probability, and so forth) require tuning for extended benefits of the algorithm to be realised.

The evolutionary approach has proved particularly successful in problems that are difficult to formalise mathematically, and which are therefore not conducive to analysis. This includes systems that are highly non-linear, that are stochastic, and that are poorly understood (control of which represents a 'black art'). Problems involving the aforementioned classes of process tend to be difficult to solve satisfactorily using conventional methods. The GA's lack of reliance on domain-specific heuristics makes it a strong contender for application in this area. Very little *a priori* information is required, but this can be incorporated if so desired.

A single control engineering problem can contain a mixture of decision variable formats. This can prove significantly problematic for conventional optimisation techniques that require variables of a single mathematical type or scientific unit. Since the GA operates on an encoding of the parameter set, diverse types of variable can be represented (and subsequently manipulated) within a single solution. For example, the decision vector {blue, 18.3°, 1+j, 2π} does not pose an intrinsic problem to the GA.

The GA is a robust search and optimisation method. It is well able to cope with ill-behaved cost landscapes, exhibiting such properties as multimodality, discontinuity, time-variance, randomness, and noise. Each of these properties can cause severe difficulties to traditional computational search methods, in addition to the lack of amenity to an analytical solution. Furthermore, a GA search is *directed* and, hence, represents potentially much greater efficiency than a totally random or enumerative search.

One particularly promising avenue of GA application is the multiobjective problem. Many real-world applications fit into this category, including most engineering design problems. The potential of the GA is only starting to be fulfilled in this arena. GAs are also capable of supporting multiple, contrasting, solutions to a problem simultaneously. This provides the designer with a greater degree of choice and flexibility.

For problems that are well understood, that are approximately linear, and for which trusted solutions exist, the GA is unlikely to produce competitive results. If a problem can be solved analytically with an acceptable level of assumptions then that approach is probably best. If such a solution cannot be found, and other problem-specific techniques are at an embryo stage, then the use of a GA could prove to be highly profitable and worthwhile.

Mission-critical and safety-critical applications do not appear, initially, to be favourable towards GA usage. There is an element of chance in a genetic algorithm. No guarantee is provided that the results will be of sufficient quality for use on-line. When GAs are evaluated

on benchmark problems, they are commonly tested many (typically twenty to thirty) times due to the stochastic nature of the algorithm. There is also the question over how individuals will be evaluated if no process model is available (as may well be the case). Some supporting theory exists for genetic algorithms, but is unlikely to prove sufficient to win the approval of standards agencies. Much care would, clearly, be needed for critical systems.

GAs are very computationally intensive, often requiring massively parallel implementations in order to produce results within an acceptable timeframe. Hence, on-line application to real-time control is largely infeasible at present.

3.2. Representation

The typical control problem contains various attributes that a GA must account for in some way. These include representation of design parameters, inclusion of constraints, assessment of performance, and methods for coping with the likely properties of the fitness landscape.

Each solution to a problem consists of a number of design parameters (decision variables). These are encoded as a chromosome, which can then be manipulated by genetic operators. Thus, a mapping is required between the actual decision variables (*phenotypes*) and their genetic equivalent (*genotypes*). Choice of representation is very flexible; indeed any is acceptable so long as suitable genetic operators can be developed to support the representation. The classic approach is to encode the set of parameters in a concatenated binary string. A simple example is shown in Figure 9.

Figure 9: Mapping between GA chromosome and decision variables

Both discrete and continuous variables can be catered for. A binary, or *n*-ary, coding can be used to represent variables to an arbitrary resolution. It should be noted, however, that the size of the search space increases exponentially with chromosome length. *Gray* encoding is often preferred to standard binary since it maintains the closeness of different solutions in both genotypic and phenotypic space. In recent years, floating-point representations have become popular alternatives to *n*-ary codings [Michalewicz, 1996]. This development has been motivated in the main by users' comfort with one-gene-one-variable correspondence, but float-encoding has several benefits: it is faster to manipulate, it has been shown empirically to be more consistent from run to run, it permits much higher precision, and it is intuitively closer to the problem space. This final point is particularly beneficial since it enables easier incorporation of domain-specific knowledge. It should be noted that the existent theory to support the effectiveness of GAs (the so-called *schema theorem*) is based on binary encoding of decision variables [Goldberg, 1989].

Most engineering problems involve constraints, including the necessity of a stable system, and actuator performance limits. Three methods have been proposed to deal with constraints within a GA solution. The most efficient approach is to embed the constraints in the genetic code, making it impossible to generate infeasible solutions. Unfortunately it may be impossible, or far from obvious how, to adopt this method. In this case, penalty functions can be incorporated. These assign a very high cost (or, correspondingly, a very low fitness) to infeasible solutions. This approach can be rather ungainly but has been sufficiently effective in many applications. The third technique, which may become more popular as MOEA research expands, is to recast the constraints as objectives to be met and, consequently, solved as a multiobjective optimisation problem.

The method by which potential solutions are assessed is a critical component of a GA. If the evaluation functions are inappropriate then the GA search is unlikely to progress in an

acceptable direction. Initially, the raw performance measures (objectives) must be defined. This is very much an application-dependent process. The next consideration is the means by which the performance data will be obtained; this is often model-based. The raw performance must then be translated into a non-negative, scalar, fitness value. The fitness value represents the expected number of times that an individual will be selected for reproduction. Many engineering problems consist of multiple objectives. For the purposes of the GA, these must be combined to form a single value. The weighted-sum approach has proved popular in the literature, since it is amenable to a solution by conventional GA methods, but Pareto-based techniques are likely to surpass this in the future.

Traditionally, the root-mean-square (RMS) of the error between the desired output and the actual output is taken as the cost of a particular solution. This measure is amenable to conventional search techniques. No such restrictions apply to the GA, so this limiting approach is no longer necessary. Having said this, many GA applications have retained this redundant measure, presumably out of force of habit.

Real-time performance is of particular interest to the engineer. GAs are notorious for the computational resources they require, although thoughtful implementations can reduce these requirements significantly. Parallel computing systems must be exploited for GAs to be used in real-time. Processes with long time constants represent the most feasible application.

Many engineering problems have the attribute of multimodality. This can cause premature convergence for conventional search techniques. GAs are not immune to this problem, although the parallel nature of the search process is a significant boon. Various GA add-ons have been developed to counter difficulties associated with multimodal cost landscapes, including adaptive genetic search operators, fitness sharing, mating restriction, and random injection (random chromosomes inserted into the new generation).

3.3. Available tools

There is no 'industry standard' GA toolbox available. Furthermore, no major commercial computer-aided engineering (CAE) package manufacturers have implemented GAs as part of their optimisation suites. However, several toolboxes are available in both the commercial and freeware sectors. Internet links to the toolboxes, together with brief summaries of their features, are provided in the Appendix.

Several GA toolboxes have been developed for the technical computing package MATLAB. GA source code is also available for several programming languages, including C++, Java, and FORTRAN. GA tools can further be found for spreadsheets, including Microsoft Excel.

4. Design Applications

Genetic algorithms have been most widely and successfully applied to off-line design applications. In the field of control systems engineering, these applications include controller design, model identification, robust stability analysis, and fault diagnosis. In some instances, GAs have been used as the sole means of design. In others, they have been combined with existing methods. In further cases, they have been combined with other *intelligent* or *metaheuristic* techniques. An *intelligent system* can make appropriate, autonomous, decisions and generally incorporates a process of learning (although no firm definition of such a system exists). Any technique that discovers new solutions, based upon experience gained from previous solutions, can be classified as a metaheuristic method.

4.1. Controllers

Genetic algorithms, and other evolutionary algorithms such as genetic programming, have been extensively applied to the off-line design of controllers. GAs have been used to obtain controller parameters (for various classes of controller), or controller structure, or occasionally both. They have also been combined with fuzzy and neural controllers to form an *intelligent control* scheme.

4.1.1. Parameter optimisation

In the early 1990s, GAs were first investigated as an alternative means of tuning PID (Proportional-Integral-Derivative) controllers. Oliveira *et al* [1991] used a standard GA to determine initial estimates for the values of PID parameters. They applied their methodology to a variety of classes of linear time-invariant (LTI) system, encompassing minimum-phase, non-minimum phase, and unstable systems. They improved the efficiency of their algorithm by identifying ancestral (already-assessed) chromosomes and avoiding re-evaluation of these. Wang and Kwok [1992] tailored a GA using inversion and preselection 'micro-operators' to PID controller tuning. They stressed the benefit of flexibility with regard to cost function (there being no requirement for mathematical tractability) and, in particular, alluded to the concept of Pareto-optimality (providing the potential to simultaneously address multiple objectives, such as transient performance and disturbance rejection). In an independent enquiry, Porter and Jones [1992] proposed a GA-based technique as a simple, generic, method of tuning digital PID controllers.

More recently, Vlachos *et al* [1999] applied a GA to the tuning of decentralised PI (Proportional-Integral) controllers for multivariable processes. Controller performance was defined in terms of time-domain bounds on the closed-loop responses for both reference following and loop interactions. This approach afforded good visualisation of the performance of potential solutions.

Onnen *et al* [1997] applied GAs to the determination of an optimal control sequence in model-based predictive control (MBPC). Particular attention was paid to MBPC for non-linear systems with input constraints. Specialised genetic coding and operators were developed, with the aim of preventing the generation of infeasible solutions. The resulting scheme was applied to a simulated batch-fed fermenter, with favourable results reported (compared to the traditional *branch-and-bound* method) for long control horizons.

A further approach to controller design using GAs is to apply the methodology *indirectly*. In such a scheme, the GA manipulates input parameters to an established controller design process, which in turn produces the final controller. The linear quadratic Gaussian (LQG) method and the H-infinity control scheme have both been utilised in this manner.

In LQG design, a GA can be used to search for the best values for the weighting factor matrix to meet design specifications, since manual selection of the matrix elements is not straightforward. For example, Mei and Goodall [2000] considered control strategies for the active steering of solid axle railway vehicles using the LQG method. LQG guarantees stability and, hence, the design procedure concentrated on obtaining acceptable performance for curving and ride quality, between which a trade-off is known to exist.

A similar search approach may be used in conjunction with an H-infinity design procedure. Here, a GA can search for pre- and post-plant weighting functions to ensure good closed-loop performance, whilst a robust controller is guaranteed as a result of the H-infinity design. Both of these *indirect* or hybrid design approaches (LQG and H-infinity) have been extended to simultaneously address multiple design objectives, achieved via the incorporation of a multiobjective genetic algorithm (MOGA). In one such example, Dakev *et al* [1997] applied a MOGA to the indirect H-infinity design of an electromagnetic suspension (EMS) control system for a maglev vehicle. This is a critical, inherently unstable, system that requires active control. Various performance criteria were optimised and compared simultaneously. The approach permitted good visualisation of the design trade-offs, such as exists between the air gap and the passenger cabin acceleration, underpinned by a robust H-infinity controller for all alternative designs.

GAs have also been successfully applied *directly* in the field of H-infinity control. In this approach, the actual controller is designed via a GA. Recognising the difficulty in implementing unconstrained H-infinity controllers, in which the order of the controller is much higher than that of the plant, Chen and Cheng [1998] proposed a structure specified H-infinity controller. A GA was used to search for good solutions within the admissible domain of controller parameters (obtained via the Routh-Hurwitz stability criterion).

It will be apparent that MOGAs have been used in a variety of parameter optimisation problems to great effect. Multiple design objectives may be defined, in both the time- and frequency-domains, resulting in a vector objective function. Progressive articulation of designer preferences can then be enabled by the use of goals and priorities. In the most popular of the MOEA approaches, Pareto ranking is used, together with niche formation techniques. In one such study, Fonseca and Fleming [1998] applied a MOGA to the optimisation of the low-pressure spool speed governor of a *Rolls-Royce Pegasus* gas turbine engine. The results highlighted the importance of *progressive* preference articulation and *interaction* with the designer, since only a small proportion of the nondominated set was found to be of practical relevance.

4.1.2. Structure

Many GA applications simply optimise the parameters of existing controller structures. Hence, they are often regarded as a direct replacement for, often tried-and-trusted, existing methods, and are used primarily in the face of ill-behaved cost landscapes. In order to harvest the full potential of the GA, some researchers have experimented with the manipulation of controller structures.

Genetic programming has been utilised for the automatic synthesis of the parameter values and the topology of controllers [Koza *et al* 2000]. A toroidal island model of 66 GP algorithms, each with 1000 individuals, has been implemented. This approach is very computationally intensive in comparison to most EAs. The system has reportedly duplicated existing patents (for PI and PID controllers) and rediscovered old ones (a controller making use of the second derivative of the error between the reference signal and the output signal).

Multiobjective evolutionary algorithms have been utilised in the context of controller structure optimisation. For example, MOGA has been used to select controller structure and

suitable parameters for a multivariable control system for a gas turbine engine [Chipperfield and Fleming, 1996]. The chromosomes contained structure genes (indexed to four pre-compensator structures) and real-coded parameters. A Breeder Genetic Algorithm (BGA) [Mühlenbein and Schlierkamp-Voosen, 1993] was used as the MOGA search engine to simultaneously optimise rise-time, settling-time, overshoot, cross-coupling, and controller complexity objectives. The family of Pareto-optimal solutions, which develop over the course of a single run, empower the engineer to examine trade-offs between design objectives and configurations.

4.1.3. Application to fuzzy / neural control

The limitations of conventional controllers for application to complicated, dynamical, systems have motivated research into so-called *intelligent control systems*. The two most popular techniques are fuzzy control, in which expert knowledge can be incorporated into the design, and neural control, which is most suitable for poorly modelled and non-linear systems. Linkens and Nyongesa [1996] present a comprehensive appraisal of both approaches.

Genetic algorithms have been used in attempts to optimise various aspects of intelligent controllers. In fuzzy control, a GA can be used to generate the fuzzy rulebase, and to tune the associated membership function parameters. In neural control, a GA can function as an alternative choice to learning the weight values. GAs have also been mooted as capable of optimising the topology of a neural network. Various illustrative examples of the application of GAs to intelligent control are presented below.

Ichikawa and Sawa [1992] used a neural network (NN) as a direct replacement for a conventional controller. The weights were obtained using a GA approach. Each individual in the population represented a weight distribution for the network. The structure and activation function were decided *a priori*. This approach offered the benefit that teaching patterns were not required and the objective function was not required to be mathematically well-behaved.

Angeline *et al* [1994] attempted the simultaneous evolution of neural network structure and weights using evolutionary programming (EP). They argued that a GA was not suitable for the task because the crossover operator is likely to act in a destructive, rather than the typically constructive, manner. Because of the distributed nature of NN processing, components from two separate well-performing networks are unlikely to perform well when fused together, even for the case when the topologies are similar. EP uses only a mutation operator to manipulate chromosomes. Therefore, the integrity of an individual network would be maintained. The EP approach also facilitates direct manipulation of the individual networks. Angeline *et al* distinguished between parametric and structural mutation operators, and applied more severe mutations to poorly performing individuals.

Work in the area of GAs applied to fuzzy control is broadly split into two categories: tuning of the membership functions, and elicitation of the rulebase in addition to tuning. Practitioners of the former approach tend to argue that the form of the rules is likely to be known *a priori*, and that most uncertainty lies in the development of the associated membership functions. Use of a static rulebase also reduces the necessary level of computational complexity, which may be a further reason for the popularity of this approach.

Tzes *et al* [1998] applied a GA to the off-line tuning of Gaussian membership functions. Using a *fuzzy clustering* technique, they developed a fuzzy model that describes the friction in a dc-motor system. The GA was seeded so that the initial genes in the pool lay close to those obtained by fuzzy clustering. Improved results were demonstrated over the non-tuned version. An asynchronous GA (in which the generations are not synchronised) has been used to optimise membership functions in order to facilitate rapid prototyping of fuzzy controllers [Kim *et al*, 1995]. This approach was suited to massively parallel processing and has been

implemented on a 512 processor CM-5 Connection Machine. The technique was applied to a simulated space-based oxygen production system.

Evolutionary methods have also been applied to the generation of control rules in situations where a reasonable set of rules is not immediately apparent. Here, the designer may either use a pre-specified number of rules or allow these to be free (thereby invoking a GP-type approach). Note that the latter technique tends to be particularly computer-intensive. Matsuura *et al* [1996] used a GA to obtain optimal control of sensory evaluation of the saké mashing process. The GA learned rules for a *fuzzy inference* mechanism, which subsequently generated the reference trajectory for a PI controller based on the sensory evaluation. GAs have also been used in the development of other types of rule bases, such as bang-bang control applied to the classic inverted pendulum control problem [Varšek *et al*, 1993].

Linkens and Nyongesa [1995] consider both the on-line (see Section 5) and off-line application of GAs to fuzzy control. They consider the complete process of fuzzy design, including elicitation of control rules and optimisation of membership functions. In addition, they provide a comprehensive discussion of GA operators and parameters from the perspective of fuzzy control.

4.2. Identification

Many control system applications, such as model-based predictive control, require some form of mathematical model of the process to be controlled. In some instances, accurate models can be derived analytically through consideration of known physical processes. This approach is often appropriate for linear, deterministic, time-invariant, single-input single-output systems, where sufficient knowledge of the system processes is available. Most real-world systems, however, do not fit into this category. In particular, they are often non-linear and poorly understood. Black-box modelling, commonly known as *system identification*, is often the only realistic approach available. In this case, input-output data from the system is used to generate a mathematical relationship between input and output.

System identification can be decomposed into two, inter-related, problems:

- *Selection* of a suitable model structure, and subsequent
- *Estimation* of model parameters.

Well-developed techniques exist for parameter estimation of linear models and linear-in-the-parameters non-linear models. Techniques for the selection of structure and for non-linear-in-the-parameters estimation are still the subject of ongoing research.

The application of GAs to black-box and grey-box model identification has received considerable interest since Kristinsson and Dumont's seminal paper in 1992. They applied GAs to the system identification of both continuous- and discrete-time systems. The technique employed can be used in on-line as well as off-line applications. The GA was used to directly identify poles and zeros, or to obtain the values of physical parameters. The cost function used was the error between the estimated and actual output over a window of data, which consisted of the current input-output pair and the previous 30 samples. For each sample point, the population was evolved for a further three generations. Kristinsson and Dumont reported comparable or better results to well-known techniques, but noted the high computational expense incurred.

Subsequent evolutionary system identification applications have attempted to optimise model parameters, or model structure, or sometimes both simultaneously.

4.2.1. Model structure

One of the central problems in system identification is the choice of the input, output, and delay terms that are to contribute to the model. GAs provide a simple method for searching the structure space for terms that make the most significant contributions to process output.

Nonlinear model term selection for NARMAX (Nonlinear AutoRegressive Moving Average eXogenous) models [Leontaritis and Billings, 1985] has been performed using a GA [Fonseca *et al*, 1993]. The problem can be cast as one of subset selection, where each individual in the population represents a specific term that might be used in the model. Ten terms is usually regarded as an acceptable number to capture the system behaviour. Each gene is associated with a particular term in a look-up table. Genetic operators should be carefully chosen to eliminate redundancy within an individual. Fonseca *et al* improved the effectiveness of the mutation operator by making the probability of mutation of a gene dependent on the quality of the associated term, measured as the variance of the residual of the model when the term is discarded. This special mutation operator has been shown, empirically, to improve the search speed. The parameters of the models can be estimated using standard least squares methods. This approach is possible because NARMAX models are linear-in-the-parameters. In a further example, Luh and Wu [1999] used migration-based GAs to identify NARX (Nonlinear AutoRegressive eXogenous) models.

Genetic programming has established itself as a popular technique for evolutionary system identification. Gray *et al* [1998] performed non-linear model structure identification using GP. They considered two representations: block diagrams (using SIMULINK) and equations (differential and integro-differential). A function library was constructed, which included basic linear and non-linear functions and also specific *a priori* knowledge. The necessary parameters for each structure can be found using an existing technique. Less orthodox methods will be required for more complicated model structures. In the case of Gray *et al*, parameters were found using a combined simulated annealing (SA) / Nelder-simplex method. The resulting scheme was applied to diverse systems of varying complexity, including simple transfer functions, a coupled water tank, and a helicopter rotor speed controller and engine.

Marenbach *et al* [1997] developed a general methodology for structure identification using GP with a block diagram library. They utilised an evaluation function that incorporated measures of both solution accuracy and block diagram complexity. Each block in the library was assigned a static value, representing the (subjective) complexity of the block. Typical blocks included time delays, switches, loops, and domain-specific elements. The developed methodology was applied to a biotechnological fed-batch fermentation process, providing a transparent insight into the structure of the process. This transparency could not be achieved using NNs, highlighting an oft-cited weakness of the connectionist approach.

Multiobjective NARMAX structure identification can be accomplished using a *multiobjective genetic programming* (MOGP) strategy, built on a MOGA platform [Rodriguez-Vázquez *et al*, 1997]. Seven objectives were optimised simultaneously: the number of model terms, model degree, model lag, residual variance, long-term prediction error, the auto-correlation function of the residuals, and the cross-correlation between the input and the residuals. These latter two objectives were defined as hard constraints. The terminal set consisted of linear input and output terms. The function set consisted of, simply, sum and product. These two functions are sufficient to construct any NARMAX model.

The key advantage of GP is the ability to incorporate domain-specific knowledge in a straightforward fashion. Also, the results can be readily understood and manipulated by the designer. However, GP structures can become complicated, and may involve redundant pathways. Minimising the complexity of a solution should normally be a specific objective. Furthermore, GP can be quite processor intensive, especially for structural identification

where a parameter estimation procedure must be carried out for *each* individual structure at *each* generation. Due to the complexity of the structure, traditional (trusted and efficient) parameter estimation methods are often impossible to apply.

4.2.2. Model parameters

In the cases where the identified model is linear-in-the-parameters, standard least-squares techniques can be used to obtain good estimates of the model's parameters. In circumstances where this is not the case, such as for non-linear rational models, other techniques may provide superior results. EA-based methodologies have been investigated as potential solutions.

Choi *et al* [2000] used an *evolution strategies* algorithm to identify the parameters of static (Karnopp) and dynamic (LuGre) friction models. The model structures were predefined (based on existing results in the literature). The results were used in a friction compensation control system, which utilised a sliding-mode controller.

4.2.3. Simultaneous optimisation of structure and parameters

Billings and Mao [1998] applied GAs to non-linear rational model identification. Both structure and parameter information was encoded in each individual to facilitate simultaneous optimisation of both elements. Rational models are not linear-in-the-parameters and, hence, the parameters cannot be accurately estimated by standard methods. The model can be manipulated to ensure that it *is* linear-in-the-parameters, but this introduces severe noise problems. The authors discovered that their approach found the correct structure and good parameter estimations for small systems, but failed to converge for more complicated (real-world) systems.

4.3. Fault diagnosis

Fault diagnosis systems have arisen from the general demand for safer and more reliable systems. The tasks of a fault diagnosis system can be split into three areas, namely:

- *Detection* of the presence of a fault,
- *Isolation* of the fault, and
- *Identification* or classification of the fault.

FDI (fault detection and isolation) is the term commonly given to the monitoring of faults in a feedback control system.

One particular facet of this research arena is *fault-tolerant control*, in which suitable control action can be generated in the presence of certain fault conditions.

A popular approach to FDI involves the generation and analysis of *residuals*. These are signals that reflect inconsistencies between nominal and faulty operation. Model-based observers are commonly used for residual generation. The competing effects of faults and modelling uncertainty (and disturbances) represent the central challenge to this technique. Hard or sudden faults will generally have a large effect, larger than the effects due to uncertainty, on the diagnostic residual. Hence, simple thresholding can be used to detect a fault condition. However, incipient faults may have a lower response than that due to uncertainty. In this case, thresholding cannot be applied directly. In summary, a trade-off exists between sensitivity of the residual to faults and robustness to modelling uncertainty.

The *disturbance decoupling* concept has been proposed as a solution to the trade-off, but this may not prove sufficient. Other solutions are required in cases where there is a lack of design freedom or data is unavailable concerning the distribution of disturbances (a fundamental requirement of disturbance decoupling). GAs have been used to optimise the design of model-based observers for residual generation.

Patton *et al* [1997] (see also Chen *et al* [1995]) formulated model-based FDI as a multiobjective optimisation problem, in which the task was to maximise the effect of faults on the residual, whilst minimising the effect of uncertainty. They formulated an overall cost function using the method of inequalities and optimised this using a GA. The approach was applied to the detection of sensor faults in a flight control system. Robust observers for fault detection have also been designed using a Pareto-based approach, in which the ranking of an individual solution is based on the number of solutions by which it is dominated [Kowalczuk *et al*, 1999]. In both these techniques, the use of a GA permitted the straightforward inclusion of various performance criteria (including previously-unused frequency-domain information).

GAs have also been applied to FDI methods that are not based on the concept of residuals. Marcu *et al* [1997] formulated the model-based diagnosis of process faults as a feature selection and classifier design problem. A multiobjective evolutionary algorithm was used for both off-line learning of regions corresponding to *known* fault and fault-free conditions (using component shapes), and for on-line identification of process coefficients (the so-called *symptom vector*).

As an alternative method of fault diagnosis, a GA can be applied to the generalised task of determining the correct problem (fault) from a collection of problems given a set of symptoms that indicate that a problem exists [Miller *et al*, 1993]. In this approach, the *relative likelihood* of a diagnosis given observable manifestations was considered. The method relied upon the availability of *a priori* probabilities that a particular disorder caused a particular symptom.

Painton and Campbell [1995] developed a technique for improving overall system reliability by considering component-level choices. For each possible component, a failure rate distribution and a cost were defined. A GA was used to search the component-choice landscape for the most reliable system, in terms of mean time-before-failure (MTBF), within a pre-defined cost ceiling. The MTBF distribution was obtained for each population member by conducting 200 trial runs (using Latin hypercube sampling). The GA was found to be highly preferable to the basic alternative: an exhaustive search.

4.4. System Analysis

Genetic algorithms have been utilised in the context of efficient robust control system design [Marrison and Stengel, 1997]. In stochastic robustness analysis, the probability that a given closed-loop system will exhibit unacceptable performance, in the presence of possible parameter variations, is evaluated. A GA can be used to manipulate a population of points in design space (each of which corresponds to the design vector of a compensator). Each design vector can be evaluated using Monte Carlo Evaluation (MCE). Marrison and Stengel used statistical tools to compare two designs and to avoid computing more MCEs than were necessary. The comparison of designs involved the determination of a statistically significant difference between the two alternatives. This result was suitable for use in the GA's tournament selection algorithm.

Methods for improving the speed of the robust design procedure have been considered [Schubert and Stengel, 1998]. This essentially involved the careful choice of a parallel computing architecture. A particular problem is the stochastic load imbalance, caused by variations in the amount of time needed to compute different MCEs. The problem was solved by the incorporation of a dynamic scheduler.

Robust stability analysis of discrete-time systems can be performed by means of an evolutionary search for system poles that lie outside the unit circle [Fadali *et al* 1999]. This method tests a sufficient condition for instability in LTI, discrete, uncertain systems with nonlinear polynomial structures (dependencies between the various parameters). Note that the

system can be shown to be *un*stable, but not stable, using this method. Fadali *et al* varied the population size and mutation probability during a run in an effort to obtain a balance between exploration and exploitation.

4.5. Robotics

Genetic algorithms have been utilised in robotics for both path planning and the design of behavioural controllers. Rana and Zalzala [1997] applied a GA to the collision-free path planning of robot arms. Each chromosome consisted of a floating-point vector representation of via points (between each end of the path). The actual path was then computed by fitting cubic splines to the points. The cost function was a weighted sum of the path length, the number of collisions, and the distribution of via points.

Genetic algorithms have also been used to improve the *dexterity* of robot manipulators [Erkmen *et al*, 2000]. Dexterity can be defined as the capability to manipulate objects in crowded, changing, and partially known environments and the ability to recover from failing grasps through a re-grasping procedure. The pre-shaping and grasping procedure required for robotic manipulators (such as *Anthrobot III*, which is analogous to the human hand) exhibits a search space consisting of unconnected feasible regions, multiple extrema, and non-linear, non-convex constraints. A GA was used to minimise a weighted sum of the following criteria:

- the positional error of each finger with respect to the final contact points,
- the manipulability and stability errors, and
- the number of collisions between fingers.

Re-grasping was achieved by perturbing the converged population associated with the optimal pre-shape, leading to convergence in a new area of the search space.

Dorigo and Columbetti [1994] used reinforcement learning to *shape* a robot to perform predefined target behaviour. Learning *classifier systems* performed the behavioural control. A GA was used as a rule discovery algorithm to generate the classifiers.

4.6. Further control-related combinatorial problems

Combinatorial problems are those in which an optimum set of choices must be made from a, usually significantly large, pool of potential choices. Commonly, analytical solutions do not exist, or exist only for simplified, textbook cases. Hence, research has focused on the development of computational methods. However, these problems can be computationally demanding because the search space tends to explode in size as the number of choices to be made increases. GAs have been found to be a competitive approach, in terms of both convenience and quality of solution. Notice should be taken here of the *No Free Lunch Theorem* [Wolpert and Macready, 1995]: if an application-specific algorithm is developed, it is likely to exhibit superior performance to a GA, but only for that particular application or subset of applications.

Many control problems are combinatorial, or have an element thereof, and can be generally viewed as a set of decisions, each with its own set of possible choices. Hence, a solution to a problem can be defined as a set of decision-choice pairs. The effects of the choices on the output may be non-separable, and dependencies may exist between certain decision-choice pairs. Various combinatorial problems can be identified in the earlier discussions, including selection of model structure, selection of controller structure, and system component selection. A further example is that of sensor and actuator placement.

Krishnakumar *et al* [1994] investigated the simultaneous optimisation of actuator-sensor placement and feedback controller gains as a means of providing optimal structural control. They aimed to discover several distinctly different solutions to present to the designer, and to this end used a GA with phenotypic sharing to promote niche formation. The evaluation function was the time-averaged control energy required to minimise the structural response to

a pre-defined random disturbance. A similar function was used by Zhang *et al* [2000] in the design of piezoelectric vibration control systems for flexible structures. Their float-encoded GA was tested on benchmark problems before being applied to a cantilever beam, a representative component of many flexible aerospace structures. Sensors and actuators were co-located to avoid possible instability arising from dis-location. Favourable results were reported compared to the existing quasi-Newton approach. Note that variations in the actuators and sensors could include types and quantities, in addition to locations.

Many combinatorial problems exist in the field of manufacturing optimisation. These include scheduling problems (job-shop, flow-shop, and dynamic), process planning, cellular manufacturing, assembly lines, product design, machine failure and maintenance, and quality control. See Dimopoulos and Zalzala [2000] for a broad review of evolutionary algorithms in manufacturing optimisation.

5. On-line applications

On-line applications present a particular challenge to the GA. Successful applications in this field have been somewhat limited to date. The benefits of a GA for on-line control systems engineering applications are the same as those discussed for off-line applications. However, an on-line GA approach must be used with particular caution. There are several considerations to be made. It is important that an appropriate control signal is provided at each sample instant. If unconstrained, the actions of the 'best' current individual of the GA may inflict severe consequences on the process. This is unacceptable is most applications, especially in the case of a safety- or mission-critical system.

Given that it may not be possible to apply the values represented by *any* individual in a GA population to the system, it is clear that evaluation of the complete, evolving, population cannot be performed on the actual process. The population may be evaluated using a process model, assuming that such a model exists, or performance may be *inferred* from system response to actual input signals. Inference may also be used as a mechanism for reducing processing requirements by making a number of full evaluations and then computing estimates for the remainder of the population based on these results.

In a real-time application there is a limited amount of time for which an optimiser can be executed between decision-points. Given current computing power, it is unlikely that a GA will execute to convergence within the sampling time limit of a typical control application. Hence, only a certain number of generations may be evolved. For systems with long sample times, an acceptable level of convergence may well be achieved.

In the case of a controller, an acceptable control signal must be provided at each control-point. If the GA has evolved for only a few generations then population performance may still be poor. A further complication is that the system, seen from the perspective of the optimiser, is changing over time. Thus, the evolved control signal at one instant can become totally inappropriate at the next. GAs can cope with time-varying landscapes to a certain extent, but a fresh run of the algorithm may be required. In this instance, the initial population can be seeded with previous 'good' solutions. Note that this does not guarantee fast convergence and may even lead to premature convergence.

There are three broad approaches to the use of GAs for on-line control [Linkens and Nyongesa, 1995]:

- Utilise a process *model*.
- Utilise the process *directly*.
- Permit *restricted tuning* of an existing controller.

The last approach can be used to ensure stability, when combined with some form of robust stability analysis, whilst permitting limited exploration. Local hill-climbing may prove superior to a GA if the limits are particularly restrictive. The full search potential of the GA is unlikely to be unlocked under these conditions. Refer to Linkens and Nyongesa [1995, 1996] for a detailed discussion of the issues behind intelligent systems for real-time control.

Lennon and Passino [1999] applied GAs on-line in a so-called *genetic model reference adaptive control* system. The particular application was 'base-braking'. This involves ensuring that a motor vehicle's brakes perform consistently as the driver commands. The problem includes time-varying operating conditions, which arise because of variations in the temperature of the brakes. A GA was used to evolve the gain of an existing lead-lag controller. The best individual was chosen to control the plant. Fitness was based on predicted future accuracy, by means of a braking process model. The parameters of this model were also obtained using GAs. Fixed controllers and plants were incorporated into the population as an insurance policy. *However*, assessment of the scheme was by means of simulation.

Lennon and Passino raised the issues of constrained processing time and the need for suitable inputs for identification purposes.

Genetic algorithms have also been proposed as on-line optimisers for automatic train operation (ATO) systems, running on mass rapid transit (MRT) networks with automatic train protection (ATP). Chang and Sim [1997] developed a scheme to reduce energy consumption by specifying intervals along the journey through which the train would coast (as opposed to motor). This is a complex problem, involving three objectives (punctuality, riding comfort, and energy consumption) and many variables, including interstation distances, gradient profiles, and the current operating conditions (train schedule, passenger load, and track voltages). Chang and Sim proposed the use of GAs as an alternative to the current, sub-optimal, predictive fuzzy control (PFC) scheme. Before the train departs for its destination station, a GA evolves a 'coast control table', which provides the coasting interval information and is referenced by the train at run-time. Variable length chromosomes were used and, hence, special genetic operators were required. *Simulated* performance was shown to be superior to PFC over a limited set of test problems.

Few applications have been reported on actual real-time use of GAs for control. In one such rarity, a GA has been used to tune the parameters of a PI controller in real-time for the on-line regulation of temperature in a heating system [Ahmad *et al*, 1997]. The stated objectives were to achieve the desired temperature as quickly as possible, whilst minimising overshoot. A degree of conflict exists between these two requirements and a weighted sum of the square of both regulator error and change in control action was taken as the cost function. The weights were varied over time, but in a manner that was defined *a priori*. A single generation of the GA was evaluated, using a plant model, between samples. The best solution found for that generation was allowed to control the plant. Encouraging results were presented for both time-invariant and time-varying environments. In the latter case, a new plant model was also identified at each time step. Despite reservations over whether or not the concerns highlighted earlier in this discussion have been adequately addressed, this application is one of the few where actual (rather than simulated) results have been offered.

GAs have also been utilised for the on-line tuning of controllers, *prior* to actual on-line usage of the system. In this case, the controller parameters remain static once the system is in operation. The main advantage of such a scheme is that a process model is not required at the design stage. An application that was amenable to such an approach was the tuning of a controller of a canned, electrical pump running on active magnetic bearings [Schroder *et al*, 2001]. Existing controller design techniques for this application are already in existence, but these require an involved design process and the development of accurate models. Hence, industry has tended to use conventional PID control, tuned manually on a prototype of the plant. Schroder *et al* utilised a MOGA in the development of a practical, automated, tuning procedure that led to both dramatically reduced design times and significant controller performance improvements. The MOGA was used to tune parameters for an existing controller structure, found through previous practical experience. All individuals were evaluated on the actual plant, but a careful procedure was developed to achieve a suitable level of safety and efficiency. Severe acceptance tests were also applied to enable the design of a safe and robust controller.

There is considerable scope for further applications of the GA to on-line control systems engineering. Continued advances in low-cost, high-performance computing will increase the viability of on-line GAs towards systems with shorter time constants. In addition to controller design, system identification and fault diagnosis are two rich veins of research that have yet to be mined. See Kristinsson and Dumont [1992] for initial research into on-line system identification.

6. Future perspectives

The future use of GAs in control is tied fundamentally to advances in the general field. Due to the largely generic nature of the methodology, developments that arise in control applications in one domain are likely to prove beneficial to other domains, such as finance. The reverse is also, of course, the case.

The continued progress in computer technology will permit further realisation of the GA methodology's potential. The main frustration with any GA is the large amount of computation required to formulate a good solution. The increased availability of low-cost, high-performance computing is undoubtedly of great benefit to the GA. Advancements in parallel architectures will further aid the GA's cause. In addition to increasing the effectiveness of the technique for off-line applications, computing developments should also make new on-line applications feasible for the first time. This latter point may be of particular value to adaptive control approaches.

A second key area for future development is the field of multiobjective optimisation in general, and decision-support tools in particular. The MOEA embodies an exciting opportunity to realise the full potential of the GA concept. Development of population-based, high-performance, robust, search techniques is just one facet of this discipline. The incorporation of, potentially multiple, decision-makers into the search process is also a crucial line of research. Issues involving preference articulation and design-space visualisation are still to be resolved. The MOEA is envisioned to form part of a multi-disciplinary, total design concept.

At present, MOEA research appears to be entering the period of intensive effort seen with the GA in the early 1990s. Development of various, competing, alternatives and the introduction of newly devised test suites on which to test them can already be seen. Whilst this is an interesting and informative avenue of research, development of search engine-independent techniques to aid the designer remain fundamental.

The combination of a GA search and optimisation method with complementary approaches to form hybrid algorithms is seen as vital in order to create a fully effective tool. For example, the use of a GA with a local hill-climber provides the benefit of a robust, global, search with an efficient fine-tuning mechanism. Hybrids involving such algorithms as Bayesian belief networks, intelligent agents, and ant foraging are expected to emerge in the literature. Neural network, fuzzy logic, and simulated annealing hybrids can already be found. Refer to Michalewicz and Fogel [2000] for further details.

When the GA is applied to a real-world problem, the approach is very much bespoke. Whilst the fundamentals of the algorithm remain the same, the details vary largely from application to application. This should be expected, since due to the *No Free Lunch Theorem*, if one algorithm performs better than another on a particular problem, the latter algorithm is certain to be superior on a separate problem. A completely generic algorithm cannot be realised in practice. The EA concept is generic, but the implementation is not.

It will be interesting to see if commercial off-the-shelf (COTS) GA packages will become available, and successful, in the future. Will the COTS products be entirely software based, or will hardware form part of the package? Furthermore, will successful consultancy firms emerge, which offer EA methods as a part of a decision-support contract? Convincing sceptical traditionalists is a key hurdle that the GA community must leap. This is true of proponents of intelligent systems in general. Much scepticism is quite well-placed: the GA lacks a strong theoretical foundation, applications are generally small-scale, and the algorithm suffers from an image of computational inefficiency. Would a GA solution meet the stringent

safety and quality standards required of a critical application? These are all fundamental issues to be addressed.

In summary, GA applications may divide into two main areas: off-line design-aid tools and robust on-line search and improvement algorithms. Research in both these provinces is still very much in its infancy. The rate at which the GA is applied to real-world problems is predicted to increase still further during the next few years.

References

Ahmad, M., Zhang, L., and Readle, J. C., 1997, *On-line genetic algorithm tuning of a PI controller for a heating system,* GALESIA '97 – Genetic Algorithms in Engineering Systems: Innovations and Applications, pp510-515.

Angeline, P. J., Saunders, G. M., and Pollack, J. B., 1994, *An Evolutionary Algorithm that Constructs Recurrent Neural Networks*, IEEE Transactions on Neural Networks, Vol. 5, No. 1, pp54-65, January 1994.

Baker, J. E., 1987, *Reducing bias and inefficiency in the selection algorithm*, Proceedings of the 2nd International Conference on Genetic Algorithms, Ed. Grefenstette, J. J., Lawrence Erlbaum Associates, pp14-21.

Billings, S. A. and Mao, K. Z., 1998, *Structure detection for nonlinear rational models using genetic algorithms*, International Journal of Systems Science, Vol. 29, No. 3, pp223-231.

Chang, C. S. and Sim, S. S., 1997, *Optimising train movements through coast control using genetic algorithms*, IEE Proceedings – Electric power applications, Vol. 144, No. 1, January 1997.

Channon, A. D. and Damper, R. I., 2000, *Towards the evolutionary emergence of increasingly complex advantageous behaviours*, International Journal of Systems Science, Vol. 31, No. 7, pp843-860.

Chen, B. S. and Cheng, Y. M., 1998, *A Structure-Specified H-Infinity Optimal Control Design for Practical Applications: A Genetic Approach*, IEEE Transactions on Control Systems Technology, Vol. 6, No. 6, pp707-718, November 1998.

Chen, J., Patton, R. J., and Liu, G. P., 1996, *Optimal residual design for fault diagnosis using multi-objective optimization and genetic algorithms*, International Journal of Systems Science, Vol. 27, No. 6, pp567-576.

Chipperfield, A. J. and Fleming, P. J., 1995, *Parallel genetic algorithms*, Chapter 39, Parallel and Distributed Computing Handbook (Ed. Zomaya, A.), McGraw-Hill, pp1034-1059.

Chipperfield, A. and Fleming, P., 1996, *Multiobjective Gas Turbine Engine Controller Design Using Genetic Algorithms,* IEEE Transactions on Industrial Electronics, Vol. 43, No. 5, pp1-5, October 1996.

Choi, S. H., Lee, C. O., and Cho, H. S., 2000, *Friction compensation control of an electropneumatic servovalve by using an evolutionary algorithm*, Proceedings of the Institution of Mechanical Engineers, Vol. 214, Part I, pp173-184.

Coello Coello, C. A., 2000, *Handling Preferences in Evolutionary Multiobjective Optimization: A Survey*, 2000 Congress on Evolutionary Computation, Vol. 1, pp30-37, Piscataway, New Jersey, July 2000.

Coello Coello, C. A., 1999, *A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques*, Knowledge and Information Systems. An International Journal, Vol. 1, No. 3, pp269-308, August 1999.

Dakev, N. V., Whidborne, J. F., Chipperfield, A. J., and Fleming, P. J., 1997, *Evolutionary H-infinity design of an electromagnetic suspension control system for a maglev vehicle*, Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp345-355.

Darwin, C., 1859, *The Origin of Species*, John Murray, London.

Deb, K., 1999, *Evolutionary Algorithms for Multi-criterion Optimization in Engineering Design*, Evolutionary Algorithms in Engineering and Computer Science, Chapter 8, pp135-161. John Wiley & Sons Ltd, Chichester.

Dimopoulos, C. and Zalzala, A. M. S., 2000, *Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions, and Comparisons*, IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2, pp93-113, July 2000.

Dorigo, M. and Columbetti, M., 1994, *Robot shaping: developing autonomous agents through learning*, Artificial Intelligence, Vol. 71, pp321-370.

Erkmen, I., Erkmen, A. M., and Günver, H., 2000, *Robot Hand Preshaping and Regrasping Using Genetic Algorithms*, International Journal of Robotics Research, Vol. 19, No. 9, pp857-874, September 2000.

Fadali, M. S., Zhang, Y., and Louis, S. J., 1999, *Robust Stability Analysis of Discrete-Time Systems Using Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 29, No. 5, pp503-508, September 1999.

Fonseca, C. M. and Fleming, P. J., 1993, *Genetic algorithms for multiobjective optimisation: Formulation, discussion and generalization*, Genetic Algorithms: Proceedings of the Fifth International Conference, Morgan Kaufmann, San Mateo, CA, pp416-423.

Fonseca, C. M. and Fleming, P. J., 1995, *An Overview of Evolutionary Algorithms in Multiobjective Optimization*, Evolutionary Computation, Vol. 3, No. 1, pp1-16.

Fonseca, C. M. and Fleming, P. J., 1998, *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation* and *Part II: Application Example*, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 28, No. 1, pp26-37 and pp38-47, January 1998.

Fonseca, C. M., Mendes, E. M., Fleming, P. J., and Billings, S. A., 1993, *Non-linear model term selection with genetic algorithms*, IEE/IEEE Workshop on Natural Algorithms in Signal Processing (Essex, UK), Vol. 2, pp27/1-27/8.

Fujita, K., Hirokawa, N., Akagi, S., Kitamura, S., and Yokohata, H., 1998, *Multi-objective optimal design of automotive engine using genetic algorithm*, Proceedings of DETC'98 – ASME Design Engineering Technical Conferences, Atlanta, Georgia, pp1-11.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reissue, Addison-Wesley Publishing Company.

Gray, G. J., Murray-Smith, D. J., Li, Y., Sharman, K. C., Weinbrunner, T., 1998, *Nonlinear model structure identification using genetic programming*, Control Engineering Practice, Vol. 6, pp1341-1352.

Haas, O. C. L., Burnham, K. J., and Mills, J. A., 1997, *On improving physical selectivity in the treatment of cancer: a systems modelling and optimisation approach*, Control Engineering Practice, Vol. 5, No. 12, pp 1739-1745.

Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press.

Ichikawa, Y. and Sawa, T., 1992, *Neural Network Application for Direct Feedback Controllers*, IEEE Transactions on Neural Networks, Vol. 3, No. 2, pp224-231, March 1992.

Kim, J., Moon, Y., and Zeigler, B. P., 1995, *Designing Fuzzy Net Controllers Using Genetic Algorithms*, IEEE Control Systems Magazine, Vol. 15, Iss. 3, pp62-72.

Kowalczuk, Z., Suchomski, P., and Bialaszewski, T., 1999, *Evolutionary Multi-Objective Pareto Optimisation of Diagnostic State Observers*, International Journal of Applied Mathematics and Computer Science, Vol. 9, No. 3, pp689-709.

Koza, J. R., 1992, *Genetic Programming – On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts.

Koza, J. R., Keane, M. A., Yu, J., Bennett III, F. H., Mydlowec, W., 2000, *Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming*, Genetic Programming and Evolvable Machines, Vol. 1, pp121-164.

KrishnaKumar, K., Swaminathan, R., and Montgomery, L., 1994, *Multiple Optimal Solutions for Structural Control Using Genetic Algorithms with Niching*, Journal of Guidance and Control, Vol. 17, No. 6, pp1374-1377.

Kristinsson, K. and Dumont, G. A., 1992, *System Identification and Control Using Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5, pp1033-1046, September/October 1992.

Lennon, W. K. and Passino, K. M., 1999, *Intelligent Control for Brake Systems*, IEEE Transactions on Control Systems Technology, Vol. 7, No. 2, pp188-202, March 1999.

Leontaritis, I. J. and Billings, S. A., 1985, *Input Output Parametric Models for Non-Linear Systems, Part 1: Deterministic Non-Linear Systems* and *Part 2: Stochastic Non-Linear Systems*, International Journal of Control, Vol. 41 No. 2, pp303-328 and pp329-344.

Linkens, D. A. and Nyongesa, H. O., 1995, *Genetic algorithms for fuzzy control – Part 1: Offline system development and application* and *Part 2: Online system development and application*, IEE Proceedings – Control Theory and Applications, Vol. 142, No. 3, pp161-176 and pp177-185, May 1995.

Linkens, D. A. and Nyongesa, H. O., 1996, *Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic control applications*, IEE Proceedings – Control Theory and Applications, Vol. 143, No. 4, pp367-386, July 1996.

Luh, G. C. and Wu, C. Y., 1999, *Non-linear system identification using genetic algorithms*, Proceedings of the Institution of Mechanical Engineers, Vol. 213, Part I, pp105-118.

Marcu, T., 1998, *A multiobjective evolutionary approach to pattern recognition for robust diagnosis of process faults*, SAFEPROCESS '97: Fault Detection, Supervision and Safety for Technical Processes 1997, Vols.1-3, pp1183-1188.

Marenbach, P., Bettenhausen, K. D., Freyer, S., Nieken, U., and Rettenmaier, H., 1997, *Data-driven structured modelling of a biotechnological fed-batch fermentation by mean of genetic programming*, Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp325-332.

Marrison, C. I. and Stengel, R. F., 1997, *Robust Control System Design Using Random Search and Genetic Algorithms*, IEEE Transactions on Automatic Control, Vol. 42, No. 6, pp835-839, June 1997.

Matsuura, K., Shiba, H., Hirotsune, M., and Nunokawa, Y., 1996, *Optimal control of sensory evaluation of the sake mashing process*, Journal of Process Control, Vol. 6, No. 5, pp323-326.

Mei, T. X. and Goodall, R. M., 2000, *LQG and GA solutions for active steering of railway vehicles*, IEE Proceedings – Control Theory and Applications, Vol. 147, No. 1, pp111-117, January 2000.

Michalewicz, Z., 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd Ed., Springer-Verlag, Berlin.

Michalewicz, Z. and Fogel, D. B., 2000, *How to Solve It: Modern Heuristics*, Corrected Second Printing, Springer-Verlag, Berlin.

Miller, J. A., Potter, W. D., Gandham, R. V., and Lapena, C. N., *An Evaluation of Local Improvement Operators for Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No.5, pp1340-1351, September/October 1993.

Mühlenbein, H. and Schlierkamp-Voosen, D., 1993, *Predictive Models for the Breeder Genetic Algorithm*, Evolutionary Computation, Vol. 1, No. 1, pp25-49.

Obayashi, S., Sasaki, D., Takeguchi, Y., Hirose, N., 2000, *Multiobjective Evolutionary Computation for Supersonic Wing-Shape Optimization*, IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2, July 2000, pp182-187.

Oliveira, P., Sequeira, J., and Sentieiro, J., 1991, *Selection of Controller Parameters using Genetic Algorithms*, Engineering Systems with Intelligence. Concepts, Tools, and Applications, Kluwer Academic Publishers, Dordrecht, Netherlands, pp431-438.

Onnen, C., Babuška, R., Kaymak, U., Sousa, J. M., Verbruggen, H. B., and Isermann, R., 1997, *Genetic algorithms for optimization in predictive control*, Control Engineering Practice, Vol. 5, Iss. 10, pp1363-1372.

Painton, L. and Campbell, J., 1995, *Genetic Algorithms in Optimization of System Reliability*, IEEE Transactions on Reliability, Vol. 44, No. 2, pp172-178, June 1995.

Patton, R. J., Chen, J., and Liu, G. P., 1997, *Robust fault detection of dynamical systems via genetic algorithms*, Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp357-364.

Porter, B. and Jones, A. H., 1992, *Genetic tuning of Digital PID Controllers*, Electronics Letters, Vol. 28, No. 9, pp843-844, 23rd April 1992.

Rana, A. S. and Zalzala, A. M. S., 1997, *Collision-free motion planning of multiarm robots using evolutionary algorithms*, Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp373-384.

Rodríguez-Vázguez, K., Fonseca, C. M., and Fleming P. J., 1997, *Multiobjective Genetic Programming: A Nonlinear System Identification Application*, Late Breaking Papers at the 1997 Genetic Programming Conference, pp207-212.

Schroder, P., Green, B., Grum, N., and Fleming, P. J., 2001, *On-line evolution of robust control systems: an industrial active magnetic bearing application*, Control Engineering Practice, Vol. 9, No. 1, pp37-49, January 2001.

Schubert, W. M. and Stengel, R. F., 1998, *Parallel Synthesis of Robust Control Systems*, IEEE Transactions on Control Systems Technology, Vol. 6, No. 6, pp701-706, November 1998.

Spears, W. M., De Jong, K. A., Bäck, T., Fogel, D. B., and de Garis, H., 1993, *An Overview of Evolutionary Computation*, Machine Learning: ECML-93, European Conference on Machine Learning, Lecture Notes in Artificial Intelligence, No. 667, Ed. Brazdil, P.B., pp442-459.

Swain, A. K. and Zalzala, A. M. S., 1998, *An overview of genetic programming: current trends and applications*, Technical Report No. 732, Department of Automatic Control and Systems Engineering, University of Sheffield, UK.

Tzes, A., Peng, P. Y., and Guthy, J., 1998, *Genetic-Based Fuzzy Clustering for DC-Motor Friction Identification and Compensation*, IEEE Transactions on Control Systems Technology, Vol. 6, No. 4, pp462-472, July 1998.

Varšek, A., Urbančič, T., and Fillipič, B., 1993, *Genetic Algorithms in Controller Design and Tuning*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 5, pp1330-1339, September/October 1993.

Veldhuizen, D. A. Van and Lamont, G. B., 2000, *Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art*, Evolutionary Computation, Vol. 8, No. 2, pp125-147.

Vlachos, C., Williams, D., and Gomm, J. B., 1999, *Genetic approach to decentralised PI controller tuning for multivariable processes*, IEE Proceedings – Control Theory and Applications, Vol. 146, No. 1, pp58-64, January 1999.

Wang, P. and Kwok, D. P., 1992, *Autotuning of Classical PID Controllers Using an Advanced Genetic Algorithm*, International Conference on Industrial Electronics, Control, Instrumentation and Automation (IECON 92), Vol. 3, pp1224-1229.

Wolpert, D. H. and Macready, W. G., 1995, *No Free Lunch Theorems for Search*, Technical Report SFI-TR-95-02-010, The Santa Fe Institute, New Mexico.

Zhang, H., Lennox, B., Goulding, P. R., and Leung, A. Y. T., 2000, *A float-encoded genetic algorithm technique for integrated optimization of piezoelectric actuator and sensor placement and feedback gains*, Smart Materials and Structures, Vol. 9, pp552-557.

Appendix – GA software

C++

GAlib is a C++ library of genetic algorithm components, developed by the Massachusetts Institute of Technology (MIT). It is available for download from http://lancet.mit.edu/ga/. The source code is free for any use, commercial or otherwise, and contains extensive features.

FORTRAN

David L. Carroll at the University of Illinois has developed a free *FORTRAN GA Driver*. It can be downloaded from http://www.staff.uiuc.edu/~carroll/ga.html.

Java

There are many simple GA demos on the World Wide Web (WWW) that are written as Java applets. Some of these may be of interest, but are of very limited use to the designer. However, a general GA toolkit, *The GA Playground*, is available. This can be found at http://www.aridolan.com/ga/gaa/gaa.html.

Microsoft Excel

Excel is a very popular general-purpose tool for simulations. Various GA implementations can be found in spreadsheet form, including the freeware *Genetik*. This tool is available at http://www.softseek.com/Education_and_Science/Math/Equation_Graphing_and_Solving/Review_25736_index.html.

MATLAB

Several GA toolboxes are available for the technical computing package MATLAB. The *GA Toolbox*, developed at the University of Sheffield, provides a wide-range of GA tools and is easily extensible. The toolbox is available for a modest charge: visit http://www.shef.ac.uk/uni/projects/gaipp/gatbx.html for further details. *GEATbx* is a fully commercial equivalent, based on the GA Toolbox. See http://www.geatbx.com for more information. Note that a MOGA toolbox extension has been developed for the GA Toolbox.

A freeware toolbox, *GAOT*, has been developed by North Carolina State University. It is quite basic, but provides several selection, crossover, and mutation choices. GAOT is available for download at http://www.ie.ncsu.edu/mirage/GAToolbox/gaot/. Another freeware implementation, written as a single routine by Michael B. Gordy, can be found at http://www.systemtechnik.tu-ilmenau.de/~pohlheim/EA_Matlab/genetic_maximisation_Matlab_M_Gordy.html.

Two Mathworks 'third party product' GA toolboxes are also available. *FlexTool(GA)*, developed by RKSites.com (formerly known as Flexible Intelligence Group LLC), is a modular user-interface driven tool. For further details see http://www.flextool.com. The alternative is the Genetic Search Toolbox, developed by Optimal Synthesis Incorporated (http://www.optsyn.com). This toolbox features a graphical code writer and has apparently been tested at United States government research laboratories.

Note that all the above links are valid at the time of writing.