PERGAMON

Control Engineering Practice 10 (2002) 1223–1241

CONTROL ENGINEERING PRACTICE

www.elsevier.com/locate/conengprac

Evolutionary algorithms in control systems engineering: a survey

P.J. Fleming*, R.C. Purshouse

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

Received 18 October 2001; accepted 1 March 2002

Abstract

Challenging optimisation problems, which elude acceptable solution via conventional methods, arise regularly in control systems engineering. Evolutionary algorithms (EAs) permit flexible representation of decision variables and performance evaluation and are robust to difficult search environments, leading to their widespread uptake in the control community. Significant applications are discussed in parameter and structure optimisation for controller design and model identification, in addition to fault diagnosis, reliable systems, robustness analysis, and robot control. Hybrid neural and fuzzy control schemes are also described. The important role of EAs in multiobjective optimisation is highlighted. Evolutionary advances in adaptive control and multidisciplinary design are predicted. © 2002 Published by Elsevier Science Ltd.

Keywords: Control system design; Genetic algorithms; Identification; Multiobjective optimisation; On-line control

1. Introduction

Practitioners of control systems engineering are increasingly addressing problems for which existing control theory and practice is unprepared. These problems typically require the consideration of multiple performance criteria and non-commensurable control variables. The systems involved may be difficult to formalise mathematically and may be poorly understood. Assumptions that are necessary for the application of classical methods, for example those involving system properties such as time-variance and noise, may be inappropriate. Many researchers have considered less-conventional techniques in order to find a satisfactory solution. Evolutionary algorithms have proven to be one such popular alternative.

The evolutionary algorithm (EA) is a robust search and optimisation methodology that is able to cope with ill-behaved problem domains, exhibiting attributes such as multimodality, discontinuity, time-variance, randomness, and noise. It permits a remarkable level of flexibility with regard to performance assessment and design specification. However, the EA is not without its own challenges. Much care must be taken to produce an effective search algorithm and the methodology suffers from the image of computational inefficiency. Application to real-time and safety-critical systems is highly limited at present.

This paper describes how the evolutionary algorithm methodology has been applied to control systems engineering. The suitability of the methodology is discussed, and methods for incorporating typical control problem characteristics are outlined. Particular consideration is given to the treatment of simultaneous, competing, objectives. A broad range of applications is captured under two major classifications: 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 an EA in real-time and directly influencing the operation of the system. Key application areas considered are controller design, model identification, robust stability analysis, and fault diagnosis. Rather than providing an exhaustive survey of applications of evolutionary algorithms in control systems engineering (which would include straightforward uses as an alternative optimiser), the paper attempts to provide a comprehensive account of published work that exploits the special nature of EAs in such applications. The paper closes with perspectives

*Corresponding author. Tel.: +44-114-222-5663; fax: +44-114-222-5138.

E-mail address: p.fleming@sheffield.ac.uk (P.J. Fleming).

0967-0661/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0967-0661(02)00081-3

on the future of evolutionary computing (EC), paying particular attention to issues of concern to the control engineer.

2. Evolutionary algorithms—an overview

2.1. Introduction

Evolutionary algorithms (EAs) are global, parallel, search and optimisation methods, founded on the principles of natural selection (Darwin, 1859) and population genetics (Fisher, 1930). The correctness of the EA framework as an abstraction of natural evolution has been challenged, for example by Channon and Damper (2000), but this should not be of undue concern to the engineer, who is using the algorithm for its robust search and optimisation properties. In general, any iterative, population-based approach that uses selection and random variation to generate new solutions can be regarded as an EA.

The evolutionary computing (EC) field has its origins in four landmark evolutionary approaches: *evolutionary programming* (EP) (Fogel, Owens, & Walsh, 1966), *evolution strategies* (ES) (Schwefel, 1965; Rechenberg, 1973), *genetic algorithms* (GA) (Holland, 1975), and *genetic programming* (GP) (Koza, 1992). The genetic algorithm was popularised by Goldberg (1989) and, probably as a result, the majority of control applications in the literature adopt this approach. GP is, perhaps, the next most popularly used method. That is not to say, however, that ES and EP are inferior: indeed the strengths of these approaches are increasingly being acknowledged. It is important to recognise that the boundaries between all four approaches are no longer distinct, with contemporary thinking progressing towards a more holistic EA framework (Michalewicz & Fogel, 2000). The key benefit of EC is *flexibility* and thus the designer should not feel constrained by a requirement to conform to a strict classification.

EAs 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 general algorithm is shown in Fig. 1.

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.

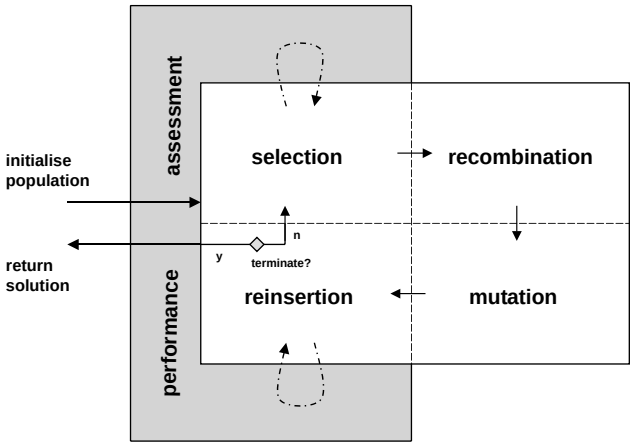

Fig. 1. Schematic of a general evolutionary algorithm.

Evolution is performed using a set of stochastic genetic operators, which manipulate the genetic code. Most evolutionary 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. *Recombination* and *mutation* are two well-known operators.

The recombination, also known as *crossover*, operator involves the exchange of genetic material between chromosomes (parents), in order to create new chromosomes (offspring). The mutation operator makes small, random, changes to a chromosome. Historically considered as a background operator by the genetic algorithm community, its rôle is often regarded as providing a guarantee that the probability of searching any given string will never be zero and providing an opportunity to recover good genetic material that may be lost through the action of selection and recombination.

Once the new generation has been constructed, the processes that result in the subsequent generation of the population are begun once more. The evolutionary algorithm explores and exploits the search space to find good solutions to the problem. It is possible for an EA to support several dissimilar, but equally good, solutions to a problem, due to its use of a population.

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

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

2.2. Fitness and cost

An evolutionary 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 EA 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 and is equally familiar to control engineers through use of such optimisation-based design procedures as the linear-quadratic regulator. 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 EA is effectively minimising the 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.

2.3. Alternative features of evolutionary algorithms

Many variations have been developed within the overall evolutionary algorithm template, with the aims of improved performance and adaptation to particular problem domains. Thus, it may be more helpful or appropriate to regard EC as a general problem-solving methodology, rather than a specific parameter-less tool. However, many applications have their basis in the *standard genetic algorithm* (SGA) presented by Goldberg (1989).

Virtually every aspect of the EA 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. Furthermore, even if a good choice of parameters can be found for a particular problem, this set will be sub-optimal for many other problems, possibly even for problems of the same general classification. Making the wrong choice of EA parameters can produce exceedingly poor results.

The specific details of each EA will vary from application to application. The choice of representation, evaluation, and genetic operators must be made with the particular application in mind. A short summary of the key issues and developments made thus far is presented below. Refer to Michalewicz and Fogel (2000) for a comprehensive discussion.

2.3.1. Representation

The EA works on an encoding of the decision variables. The classic GA choice is to encode each variable as a binary string, and then concatenate each variable to form a single solution. However, any representation is permitted for which suitable variation operators can be devised. Direct floating-point representations are becoming increasingly popular (Michalewicz, 1996). Refer to Section 3.3 for a discussion on potential representations for control problems.

2.3.2. Evaluation

The EA requires a scalar value representing the performance of each solution. This may be an absolute quantity or may be relative to other solutions. Techniques for the conversion of raw performance to an EA fitness value have received much attention in the literature. Ranking of solutions is a popular approach, especially for multiobjective problems. Individual fitness values may then be modified to encourage *niching* behaviour (the formation of sub-populations at different, comparatively optimal, locations) through the use of *fitness sharing* (Goldberg & Richardson, 1987).

2.3.3. Selection

Individual solutions must be selected for application of the genetic operators and for subsequent reinsertion into the population if a generational gap is used. 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.

2.3.4. Variation

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. Choices of values for the parameters of the operators, such as mutation rate and mutation size, are critical to the success of the algorithm. Note that parameters may vary adaptively over the course of a run in order to improve the search results and these parameters may themselves be subject to evolution (Rechenberg, 1973).

2.3.5. Population

EAs evolve a population of solutions, usually comprising a static number of between 20 and 100 individuals, over a number of generations. The population is initialised with random values, possibly seeded with some known good solutions. At each subsequent generation, the population will comprise the results of genetic manipulation, some remnants of the past population, and possibly a few randomly generated individuals. An EA is defined as *elitist* if it deliberately reintroduces past high-performance individuals. Researchers have also experimented with various population topologies. For example, sub-populations can be implemented as *islands*, which evolve in parallel, with limited migration of individuals between the islands.

The island model provides one mechanism for the incorporation of explicit parallelism within an evolutionary algorithm. For example, a single process running on a single processor could represent each sub-population. The migrants would represent data flow between the processes. Further parallel approaches such as the *farmer-worker* system and the *diffusion model* are possible. Chipperfield and Fleming (1995, Chapter 39) provide a broad overview and comparison of parallel EAs.

2.4. 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 in the two-objective case is illustrated in Fig. 2 using a simplified controller design problem in which rise time and overshoot characteristics of a step response must be simultaneously minimised.

The shaded region in Fig. 2 indicates the region of objective space for which feasible solutions exist. The trade-off surface between rise time and overshoot is described by the dark curve and lies on a part of the boundary of the feasible region (in the general n-objective case, the trade-offs will be described by a hypersurface). Any improvement to rise time on this curve is accompanied by a deterioration in overshoot performance, and vice versa. Performance vectors associated with a number of solutions are also shown in Fig. 2. The PO (non-dominated) solutions are indicated by filled circles. Each solution marked with a cross is dominated (improved on) by at least one of the PO solutions. Note that, given an arbitrary finite set of solutions, those solutions that are non-dominated with respect to the set are not necessarily PO.

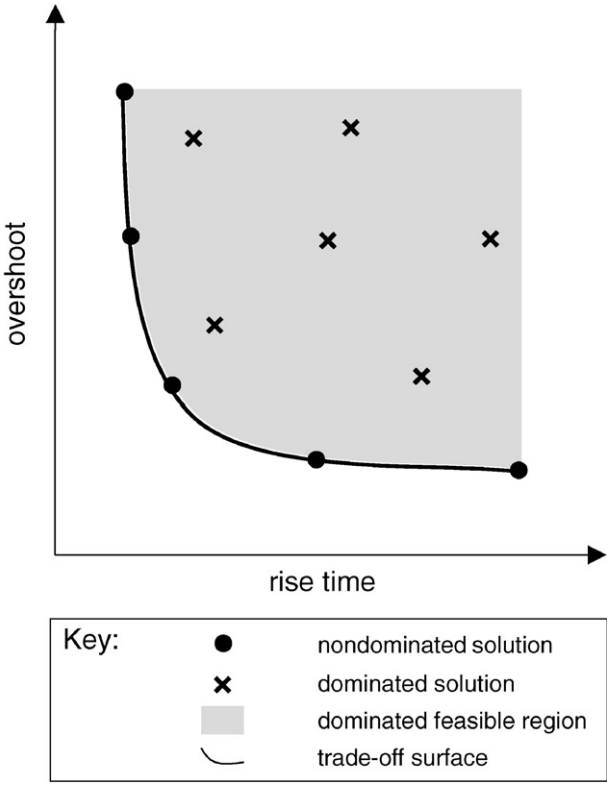

Fig. 2. Pareto optimality.

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 cost function is defined prior to the optimisation procedure. This requires in-depth information concerning the various trade-offs and valuation of each individual objective. This data is not commonly fully available in practice. Even then, such a procedure returns only a single solution (one point on the trade-off surface) per optimiser run. While multiple runs can be made with different settings for the weights, there is no guarantee that a uniform spread of objective weightings will provide a good spread of solutions. By contrast, evolutionary algorithms, due to their population-based nature, are capable of supporting several different solutions simultaneously. Thus, in a single optimiser run, the decision maker is provided with an indication of the trade-offs within the problem.

Furthermore, the weighted-sum approach is unable to identify non-convex parts of the trade-off surface, potentially missing important areas for compromise

(Fleming & Pashkevich, 1985). In contrast, the EA 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 robustness of the EA in the face of ill-behaved problem landscapes further increases the value of its utility.

Research into multiobjective evolutionary algorithms (MOEAs) 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, Chapter 8), Fonseca and Fleming (1995). A book on the subject is also now available (Deb, 2001).

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 beneficially incorporated in order to direct the search to useful parts of the trade-off surface. Incorporation of designer preferences within an 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 including *goal programming* and *lexicographic ordering*. 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, Burnham, & Mills, 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, Sasaki, Takeguchi, & Hirose, 2000) and automotive engine design (Fujita, Hirokawa, Akagi, Kitamura, & Yokohata, 1998). Thompson, Chipperfield, Fleming, and Legge (1999) consider the application of MOGA to the multidisciplinary task of control systems architecture selection. Further control-related applications are described in Section 4, many of which extend design capabilities of EA search methods based on single objectives.

3. EAs in control—benefits and shortcomings

This section describes the main features of EAs that are beneficial in control systems engineering. The benefits of EAs for use in complex problems are described, together with challenges that may limit their application. The case for using conventional approaches instead of EAs is also presented. Lastly, the methods by which standard control problem attributes, such as constraints, can be handled by an EA are discussed.

3.1. Suitability

Many exponents of evolutionary algorithms cite their generic nature as a major advantage. EAs can be applied to a wide range of problems without significant modification. However, the EA 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 paper. It should also be noted that EA 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. Problems involving these classes of process tend to be difficult to solve satisfactorily using conventional methods. The EA's lack of reliance on domain-specific heuristics makes it attractive 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 optimisers that require variables of a single mathematical type or scientific unit. Since the EA operates on an encoding of the parameter set, diverse types of variable can be represented (and subsequently manipulated in an appropriate manner) within a single solution. For example, the decision vector $\{sensor\_type\_A,\ 18.3^o,\ blue,\ 2\pi\}$ does not pose an intrinsic problem to the EA.

The EA is a robust search and optimisation method that 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, an EA search is *directed* and, hence, represents potentially much greater efficiency than a totally random or enumerative search.

One particularly promising avenue of EA application is the multiobjective problem. Many real-world applications can be categorised as such, including most engineering design problems. The potential of the EA is only starting to be realised in this area. EAs are also capable of supporting multiple, contrasting, solutions to a problem simultaneously. This provides the designer with a greater degree of choice and flexibility.

3.2. Unsuitability

For problems that are well understood, that are approximately linear, and for which trusted solutions exist, the EA 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 an EA could prove to be highly profitable and worthwhile.

Mission-critical and safety-critical applications do not appear, initially, to be favourable towards EA usage due to the stochastic nature of the evolutionary algorithm. No guarantee is provided that the results will be of sufficient quality for use on-line. When EAs are evaluated on benchmark problems, they are commonly tested many (typically 20–30) times due to the algorithms' stochastic nature. There is also the matter of how individuals will be evaluated if no process model is available (as may well be the case). Some supporting theory exists for evolutionary algorithms, especially for ES, but this is unlikely to prove sufficient to win the approval of standards and certification agencies. Much care would clearly be needed for critical systems.

Real-time performance is of particular interest to the engineer. However, EAs 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. Processes with long time-constants represent the most feasible application.

3.3. Representation: utility for control

The typical control problem contains various attributes that an EA 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.

3.3.1. Numeric representations

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 most popular approach is to encode the set of parameters in a concatenated binary string, as shown by the simple example in Fig. 3.

Fig. 3. Example mapping between 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.

3.3.2. Symbolic representations

The representation of decision variables can be symbolic rather than numeric. The evolutionary algorithm known as GP facilitates the representation and manipulation of variable length structures (Koza, 1992). The structures are composed of functions and terminals (potential inputs) that are defined in a library. For example, block diagram representations can be used. The population of block diagram structures is evolved using special genetic operators. An example of single-point binary crossover using block diagram individuals is shown in Fig. 4.

A possible GP mutation operator is illustrated in Fig. 5. In this case, each block is probabilistically tested for possible mutation. Mutation is enacted by swapping the block within the individual for another from the library of possible blocks.

3.4. Constraint handling

Most engineering problems involve constraints, including the need to have a stable system, and to observe

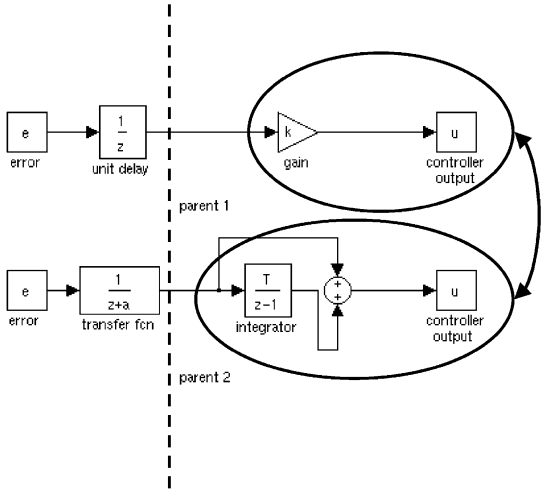

Fig. 4. Single-point crossover for GP using block diagrams.

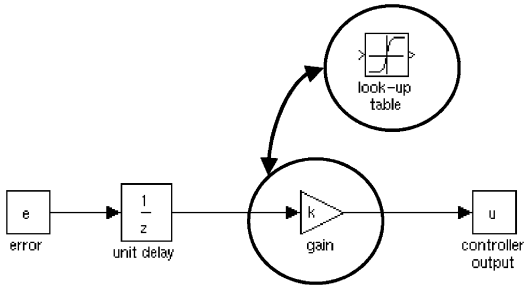

Fig. 5. Mutation for GP using block diagrams.

actuator performance limits. Five main methods have been proposed to deal with constraints within an EA solution:

(1) *Coding*: The most efficient approach is to embed the constraints in the genetic code, making it impossible to generate infeasible solutions.
(2) *Penalty functions*: Unfortunately it may be impossible, or far from obvious how, to adopt the coding 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.
(3) *Repair algorithm*: As an alternative to simply penalising infeasible solutions, a repair algorithm can be used to attempt to convert them to neighbouring feasible solutions. The fitness of these new solutions is typically reduced by an amount equivalent to the 'cost' of the associated repair.
(4) *Indirect methods of representation*: Instead of repairing the infeasible chromosome, it is also possible to reinterpret the meaning of the existing chromosome. This idea forms the basis of the indirect methods of representation, in which genotypic information passes through an interpreter that builds solutions in the phenotypic domain. In this case, it can prove very difficult to analyse the effects of genetic operators due to the complicated mapping between genotype and phenotype. However, feasible solutions are guaranteed providing that the interpreter is carefully constructed.
(5) *Multiobjective optimisation*: The final 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.

3.5. Cost functions

The method by which potential solutions are assessed is a critical component of an EA. If the cost functions are inappropriate then the EA 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 EA, these must be combined to form a single value. The (flawed) weighted-sum approach has proved popular in the literature, since it is amenable to a solution by conventional EA 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 EA, so this limiting approach is no longer necessary. Having said this, many EA applications have retained this redundant measure, presumably out of force of habit.

Many engineering problems have the attribute of multimodality. This can cause premature convergence for conventional search techniques. EAs are not immune to this problem, although the parallel nature of the search process is a significant benefit. Various means 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 in an effort to preserve diversity). Refer to Rechenberg (1973), Goldberg and Richardson (1987), Deb and Goldberg (1989),

Grefenstette (1992) for examples of seminal work on each respective topic.

4. Design applications

Evolutionary 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, system reliability, and fault diagnosis. In some instances, EAs 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 *meta-heuristic* 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 meta-heuristic method.

4.1. Controllers

GA, and other evolutionary algorithms such as GP, has been extensively applied to the off-line design of controllers. EAs 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

An early use of EAs was as an alternative means of tuning proportional-integral-derivative (PID) controllers. Oliveira, Sequeira, and Sentieiro (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 an EA 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 an EA-based technique as a simple, generic, method of tuning digital PID controllers.

More recently, Vlachos, Williams, and Gomm (1999) applied an EA to the tuning of decentralised proportional-integral (PI) 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 EAs 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 EAs is to apply the methodology *indirectly*. In such a scheme, the EA 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 a conventional LQG approach, the designer will experiment with different weighting matrices in order to achieve the performance requirements. Such manual selection of the matrix elements is not straightforward and, thus, EC can be used to automate the search for the best values for the weighting matrices that realise the design specifications. In one such 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; a trade-off is known to exist between these two objectives.

A similar search approach has been used in conjunction with an H-infinity design procedure. Here, an EA searches 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 MOGA. In one such example, Dakev, Whidborne, Chipperfield, and Fleming (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 that exists between the air gap and the passenger cabin

acceleration, underpinned by a robust H-infinity controller for all alternative designs.

EAs have also been successfully applied *directly* in the field of H-infinity control. In this approach, the actual controller is designed via an EA. 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. An EA 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 non-dominated set was found to be of practical relevance.

4.1.2. Structure

Many EA 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 EA, some researchers have experimented with the manipulation of controller structures.

GP has been utilised for the automatic synthesis of the parameter values and the topology of controllers (Koza, Keane, Yu, Bennett III, & Mydlowec, 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).

MOEAs 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 & Fleming, 1996). The chromosomes contained structure genes (indexed to four pre-compensator structures) and real-coded parameters. A breeder genetic algorithm (BGA) (Mühlenbein & 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 the 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.

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

Ichikawa and Sawa (1992) used an NN as a direct replacement for a conventional controller. The weights were obtained using an EA 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.

NN and evolutionary algorithm techniques have been combined in an attempt to optimise a cotton spinning process, in terms of both the tenacity and elongation of the yarn (Sette, Boullart, & Langenhove, 1998). The inputs to the system are five machine parameter settings and 12 characteristics of the raw material used. A multilayer perceptron neural network, trained with backpropagation, was used to learn the underlying process function. The architecture, learning factor, and smoothing factor of the NN were optimised using a GA. Subsequently, a multiobjective GA was applied to the NN model in order to produce a family of potential solutions. A real cotton fibre that best represented a selected member of the family (in the least-squares sense) was then used, together with the optimised machine configuration, to produce a new yarn.

Angeline, Saunders, and Pollack (1994) attempted the simultaneous evolution of NN structure and weights

using EP. They argued that a GA was not suitable for the task because the recombination 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 EAs applied to fuzzy control is broadly split into two categories:

(a) tuning of the membership functions, and
(b) elicitation of the rulebase in addition to tuning.

Practitioners of approach (a) 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, Peng, and Guthy (1998) applied an EA 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 EA 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 EA (in which the generations are not synchronised) has been used to optimise membership functions in order to facilitate rapid prototyping of fuzzy controllers (Kim, Moon, & Zeigler, 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, Shiba, Hirotsune, and Nunokawa (1996) used an EA to obtain optimal control of sensory evaluation of the saké mashing process. The EA learned rules for a *fuzzy inference* mechanism, which subsequently generated the reference trajectory for a PI controller based on the sensory evaluation. EAs 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, Urbančič, & Fillipič, 1993).

Linkens and Nyongesa (1995) consider both the on-line (see Section 5) and off-line application of EAs 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 EA operators and parameters from the perspective of fuzzy control.

4.2. System identification

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

- *selection* of a suitable model structure, and
- *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 EAs to black-box and grey-box model identification has received considerable interest since the seminal paper by Kristinsson and Dumont (1992). The authors 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. EAs provide a simple method for searching the structure space for terms that make the most significant contributions to process output.

Term selection for Non-linear AutoRegressive Moving Average eXogenous (NARMAX) (Leontaritis & Billings, 1985) polynomial models has been performed using an EA (Fonseca, Mendes, Fleming, & Billings,

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. An acceptable number of terms to capture the system behaviour is estimated a priori. Each gene is associated with a particular term in a look-up table. Genetic operators are 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 polynomial models are linear-in-the-parameters. In a further example, Luh and Wu (1999) used migration-based EAs to identify Nonlinear Auto-Regressive eXogenous (NARX) models.

GP has established itself as a popular technique for evolutionary system identification. Gray, Murray-Smith, Li, Sharman, and Weinbrunner (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, Bettenhausen, Freyer, Nieken, and Rettenmaier (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 polynomial model structure identification has been accomplished using a *multiobjective genetic programming* (MOGP) strategy, built on a MOGA platform (Rodríguez-Vásquez, Fonseca, & Fleming, 1997). Here, 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. The latter two objectives were defined as hard constraints. The terminal set consisted of linear input and output terms. The function set simply consisted of sum and product. These two functions are sufficient to construct any NARMAX polynomial 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; the topic of 'bloat' is a continuing area of study for GP researchers (Langdon & Poli, 1997). 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, Lee, and Cho (2000) used an ES 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 EAs 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,

commonly referred to as fault detection and isolation (FDI). 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). EAs have been used to optimise the design of model-based observers for residual generation.

Patton, Chen, and Liu (1997) (see also Chen, Patton, & Liu, 1996) formulated model-based FDI as a multi-objective 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 (Zakian & Al-Naib, 1973) 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, Suchomski, & Bialaszewski, 1999). In both these techniques, the use of an EA permitted the straightforward inclusion of various performance criteria (including previously unused frequency-domain information).

EAs have also been applied to FDI methods that are not based on the concept of residuals. Marcu (1998) formulated the model-based diagnosis of process faults as a feature selection and classifier design problem. An MOEA 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, an EA has been 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, Potter, Gandham, & Lapena, 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.

4.4. Reliable systems

Evolutionary algorithms have played a part in the drive to improve the reliability of systems. In an early study, 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. An EA was used to search the component-choice landscape for the most reliable system, in terms of mean time-before-failure (MTBF), within a predefined cost ceiling. The MTBF distribution was obtained for each population member by conducting 200 trial runs (using Latin hypercube sampling). The EA was found to be highly preferable to the basic alternative: an exhaustive search.

Coit and Smith (1996) also considered multiple component choices, but furthermore introduced redundancy by using an EA to find optimal design configurations for series–parallel systems. Each chromosome consisted of several series subsections, within which parallel components were selected from a look-up table. Performance criteria were defined as reliability, cost, and weight. Note that, in this implementation, only a single criterion was optimised at a time, the others being treated as constraints. Coit and Smith achieved good results when the technique was applied to two benchmark problems.

EAs have also been used to search for optimal design configurations using a *fault tree* approach (Ren & Bechta Dugan, 1998). The fault tree methodology provides a mathematical and graphical representation of combinations of events that can lead to system failure. Each individual in the EA can represent a set of component and redundancy choices (essentially

switching on or off different branches in an enumerated fault tree). The resulting trees can be solved for reliability using conventional methods, such as Markov chain analysis. Ren and Bechta Dugan improved the efficiency of their approach by storing results for solved branches of the tree. The cost function used was a weighted sum of failure probability, expense, and power consumption. The authors note that re-runs with different weights would be required in order to gather trade-off information. The methodology has been applied to a cardiac-assist system used to treat heart failure.

4.5. Robust stability analysis

Evolutionary algorithms have been utilised in the context of efficient robust control system design (Marrison & 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. An EA 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 EA tournament selection algorithm.

Wang and Stengel (2000) incorporated the aforementioned stochastic robustness technique with non-linear dynamic inversion in order to develop a robust non-linear controller for a hypersonic aircraft. This system contained 28 parameters that were subject to uncertainty, and 39 stability and performance requirements were defined.

Methods for improving the speed of the robust design procedure have been considered (Schubert & 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, Zhang, & Louis, 1999). This method tests a sufficient condition for instability in LTI, discrete, uncertain systems with non-linear polynomial structures (dependencies between the various parameters). Note that the system can be shown to be unstable, 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.

4.6. Robotics

Evolutionary algorithms have been utilised in robotics for both path planning and the design of behavioural controllers. Rana and Zalzala (1997) applied an EA 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.

Evolutionary algorithms have also been used to improve the *dexterity* of robot manipulators (Erkmen, Erkmen, & Günver, 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. Two underlying measures are *manipulability* (the ability to change the orientation of the grasped object) and *stability* (all contacts on the object produce compressive forces). 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. An EA 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 classifier is a cognitive system capable of recognising objects and events in its environment and of making appropriate responses). An EA was used as a rule discovery algorithm to generate the classifiers.

4.7. 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. EAs 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 & Macready, 1997): if an application-specific algorithm is developed, it is likely to exhibit superior performance to an EA, 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, Swaminathan, and Montgomery (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 an EA 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, Lennox, Goulding, and Leung (2000) in the design of piezoelectric vibration control systems for flexible structures. Their float-encoded EA 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 dislocation. 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 EA. Successful applications in this field have been somewhat limited to date. The benefits of an EA for on-line control systems engineering applications are the same as those discussed for off-line applications. However, an on-line EA 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 EA may inflict severe consequences on the process. This is unacceptable in 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 an EA 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 an EA 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 EA 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. EAs 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 EAs for on-line control (Linkens & 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 EA if the limits are particularly restrictive. The full search potential of the EA 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 EAs 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. An EA 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 EAs. 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.

Evolutionary 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 EAs as an alternative to the current, sub-optimal, predictive fuzzy control (PFC) scheme. Before the train departs for its destination station, an EA 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.

Chang and Xu (2000) proposed to optimise the original fuzzy controller for the MRT system in real-time using an EA. The membership functions for safety, punctuality, energy efficiency, and passenger comfort were tuned using a *differential evolution* (DE) algorithm (which the authors believe offers good convergence to the global optimum). Evaluation of potential solutions was achieved by means of simulated train performance. Evolved results were stored and reused to improve efficiency. Simulated results suggest that a significant improvement in performance can be obtained, but further thought is required with regard to time constraints and safety issues.

Few applications have been reported on actual real-time use of EAs for control. In one such rarity, an EA 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, Zhang, & Readle, 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 EA 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.

In another rare real-time scheme, Moriyama and Shimizu (1996) utilised an EA to determine the optimal temperature profile for an ethanol fermentation process. Each binary chromosome represented a culture temperature time series. A model was used to evaluate individuals, with the first future set-point of the best individual being used as the process set-point for an interval of thirty minutes, in an approach somewhat analogous to MBPC. The parameters of the model were updated using on-line measurements. Recognising the need to reach satisfactory convergence within the available thirty minutes, the authors determined good values for population size and the maximum number of generations off-line. The scheme was implemented on a dual processor system, with sensor–actuator processing and optimisation being handled in parallel. Actual results indicate that a 14% increase in productivity has been achieved. However, Moriyama and Shimizu do not address the danger of applying an inappropriate set-point for 30 min.

EAs 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, Green, Grum, & Fleming, 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. (2001) 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 EC to on-line control systems engineering. Continued advances in low-cost, high-performance computing will increase the viability of on-line EAs 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. Conclusions

6.1. The survey

This survey has introduced the EC methodology and has sought to highlight the special properties of EAs that will be of interest to control system engineers. The survey is not intended to be all embracing; indeed, in terms of quantity of citations it represents only a small subset of the available material. However, the applications described in this paper are indicative of the breadth of control-related topics in which EAs have proved profitable. The cited work was selected because of its seminal nature and/or the fact that it exploits the EA framework in an interesting way.

EAs are flexible design tools for control systems engineering. The design variables can be represented in a natural fashion, as can the performance specifications. The optimisation process can be tuned to a great extent to the problem at hand. EC represents a highly competitive alternative to conventional approaches in cases where these latter methods fail or require unacceptable assumptions. A significant and successful body of EA research work exists for constraint handling methods. EC is also proving to be singularly successful in effectively addressing multiobjective problems.

A common use of EAs is as off-line controller optimisers, where they have been used extensively in choosing controller parameters and searching for alternative structures. Employing EAs as "design assistants" for design parameter selection in approaches such as H-infinity and LQG has enhanced the utility of these methods. The flexibility of EC has enabled EAs to be effective tools within neural and fuzzy control schemes. They are also popular choices for model-building tasks in system identification. The possibility for a natural, symbolic, representation of the system has made the GP methodology invaluable in these cases. Many useful applications of EAs can also be found in fault detection, systems reliability, and robustness analysis.

On-line applications of EC have proved particularly problematic because of time constraints, restrictions on evaluation of candidate solutions, and fears over robustness. However some interesting and successful studies do exist, including cases of actual real-time usage.

The survey has revealed convincing examples of the utility of EA methodology in control systems engineering. Furthermore, this methodology is readily transferred into industrial use, because (a) the nature of the paradigm is attractive to practising engineers who readily appreciate the concepts involved, and (b) the versatility of the paradigm enables it to be adapted to meet the demands of difficult industrial problems.

6.2. Future perspectives

The future use of evolutionary algorithms in control is tied fundamentally to general advances in EC itself. 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 EA methodology's potential. The main frustration with any EA is the large amount of computation required to formulate a good solution. Hence, the increased availability of low-cost, high-performance computing, coupled with advancements in parallel architectures, will undoubtedly be of great benefit. In addition to increasing the effectiveness of the technique for off-line applications, computing developments should also make new on-line applications, such as evolutionary adaptive control, feasible for the first time.

A second key area for future development is the field of multiobjective optimisation and decision support. The MOEA embodies an exciting opportunity to realise the full potential of the EA concept. The development of population-based, high-performance, robust, search techniques is just one facet of this research area; of crucial importance is the incorporation of, potentially multiple, decision makers into the search process. Preference articulation and decision-space (and

criterion-space) visualisation are both issues of ongoing research.

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, the quest for a Utopian algorithm is ultimately futile (Wolpert & Macready, 1997). The development of search engine-independent techniques to aid the designer remains a fundamental requirement of MOEA research.

The combination of an EA search and optimisation method with complementary techniques to create 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. NN, fuzzy logic, and simulated annealing hybrids can already be found. Refer to Michalewicz and Fogel (2000) for further details.

When an EA 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 (Wolpert & Macready, 1997). Thus, a completely generic algorithm cannot be realised in practice; the EA *concept* is generic, but the implementation is not. In point of fact, EC is in need of a handbook, within which all EA design choices presented in the literature are gathered together, critically appraised and summarised to prevent the 'reinvention of the wheel'. This is similar to the 'design patterns' found in object-oriented design. It is important that researchers make their own decisions regarding their own problems—fitting the tool to the problem, rather than the reverse—but guidance and support is positively required.

The emergence of a standard EA toolbox has been predicted throughout the 1990s. However, despite the availability of packages developed by several research laboratories, a standard set of tools has not been adopted. Furthermore, no major design tool manufacturer has integrated an EA tool into their software package. This is perhaps largely due to the required customisation as mentioned above. Commercial off-the-shelf (COTS) EA packages are expected to become more widely available in the future. These may be entirely software based, or, perhaps, hardware will form part of the package. A possible development is the emergence of consultancies that offer EA methods as a part of a decision-support service.

The true challenge that proponents of EAs must overcome, and this is true of intelligent systems in general, is to convince sceptical traditionalists that the new methods are worthy of their trust. Much scepticism is quite well placed: the EA lacks a strong theoretical foundation, applications are generally small-scale, and the algorithm suffers from an image of computational inefficiency. Furthermore, how confident would an EA designer be that their solution would meet the stringent safety and quality standards required of a critical application? GA applications are required that *have actually been fully implemented* on the chosen real-world problem.

In summary, significant future EA applications are likely focus on two main areas: off-line, multicriterion, 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 EA is applied to real-world problems is predicted to increase still further during the next few years.

References

Ahmad, M., Zhang, L., & Readle, J. C. (1997). On-line genetic algorithm tuning of a PI controller for a heating system. *Proceedings of GALESIA '97—genetic algorithms in engineering systems: Innovations and applications*. Glasgow, UK (pp. 510–515).

Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 54–65.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the second international conference on genetic algorithms*. Cambridge, USA (pp. 14–21).

Billings, S. A., & Mao, K. Z. (1998). Structure detection for nonlinear rational models using genetic algorithms. *International Journal of Systems Science*, *29*(3), 223–231.

Chang, C. S., & Sim, S. S. (1997). Optimising train movements through coast control using genetic algorithms. *IEE Proceedings—Electric power applications*, *144*(1), 65–73.

Chang, C. S., & Xu, D. Y. (2000). Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system. *IEE Proceedings—Electric power applications*, *147*(3), 206–212.

Channon, A. D., & Damper, R. I. (2000). Towards the evolutionary emergence of increasingly complex advantageous behaviours. *International Journal of Systems Science*, *31*(7), 843–860.

Chen, B. S., & Cheng, Y. M. (1998). A structure-specified H-infinity optimal control design for practical applications: A genetic approach. *IEEE Transactions on Control Systems Technology*, *6*(6), 707–718.

Chen, J., Patton, R. J., & Liu, G. P. (1996). Optimal residual design for fault diagnosis using multi-objective optimization and genetic algorithms. *International Journal of Systems Science*, *27*(6), 567–576.

Chipperfield, A.J., Fleming, (1995). Parallel genetic algorithms. In: A. Zomaya (Ed.), *Parallel and Distributed Computing Handbook* (pp. 1034–1059). New York: McGraw-Hill.

Chipperfield, A., & Fleming, P. (1996). Multiobjective gas turbine engine controller design using genetic algorithms. *IEEE Transactions on Industrial Electronics*, *43*(5), 1–5.

Choi, S. H., Lee, C. O., & Cho, H. S. (2000). Friction compensation control of an electropneumatic servovalve by using an evolutionary algorithm. *Proceedings of the Institution of Mechanical Engineers Part I*, *214*, 173–184.

Coello, C. A. C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems. *An International Journal*, *1*(3), 269–308.

Coello, C. A. C. (2000). Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. *Proceedings of the 2000 congress on evolutionary computation*, Vol. 1. San Diego, USA (pp. 30–37).

Coit, D. W., & Smith, A. E. (1996). Reliability optimization of series–parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, *45*(2), 254–260.

Dakev, N. V., Whidborne, J. F., Chipperfield, A. J., & 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 Part I*, *211*, 345–355.

Darwin, C. (1859). *The origin of species*. London: John Murray.

Deb, K. (1999). Evolutionary algorithms for multi-criterion optimization in engineering design. In: K. Miettinen et al. (Eds.), *Evolutionary algorithms in engineering and computer science* (pp. 135–161). Chichester: Wiley.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester: Wiley.

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. *Proceedings of the third international conference on genetic algorithms*. Washington DC, USA (pp. 42–50).

Dimopoulos, C., & Zalzala, A. M. S. (2000). Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation*, *4*(2), 93–113.

Dorigo, M., & Columbetti, M. (1994). Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, *71*, 321–370.

Erkmen, I., Erkmen, A. M., & Günver, H. (2000). Robot hand preshaping and regrasping using genetic algorithms. *International Journal of Robotics Research*, *19*(9), 857–874.

Fadali, M. S., Zhang, Y., & 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*, *29*(5), 503–508.

Fisher, R. A. (1930). *The genetical theory of natural selection*. Oxford: Clarendon Press.

Fleming, P.J., & Pashkevich, A. P. (1985). Computer aided control system design using a multiobjective optimization approach. *Proceedings of the IEE control '85 conference*. Cambridge, UK (pp. 174–179).

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution.*. New York: Wiley.

Fonseca, C.M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimisation: Formulation, discussion and generalization. *Proceedings of the fifth international conference on genetic algorithms*. San Mateo, USA (pp. 416–423).

Fonseca, C. M., & Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*(1), 1–16.

Fonseca, C.M., & 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*, *28*(1), 26–37 and 38–47.

Fonseca, C.M., Mendes, E.M., Fleming, P.J., & Billings, S. A. (1993). Non-linear model term selection with genetic algorithms. *IEE/IEEE workshop on natural algorithms in signal processing*, Vol. 2. Essex, UK (pp. 27/1–27/8).

Fujita, K., Hirokawa, N., Akagi, S., Kitamura, S., & Yokohata, H. (1998). Multi-objective optimal design of automotive engine using genetic algorithm. *Proceedings of DETC'98—ASME design engineering technical conferences*. Atlanta, USA (pp. 1–11).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D.E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the second international conference on genetic algorithms*. Cambridge, USA (pp. 41–49).

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*, *6*(11), 1341–1352.

Grefenstette, J. J. (1992). Genetic algorithms for changing environments. *Proceedings of parallel problem solving from nature*, Vol. 2. Brussels, Belgium (pp. 137–144).

Haas, O. C. L., Burnham, K. J., & Mills, J. A. (1997). On improving physical selectivity in the treatment of cancer: A systems modelling and optimisation approach. *Control Engineering Practice*, *5*(12), 1739–1745.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Ichikawa, Y., & Sawa, T. (1992). Neural network application for direct feedback controllers. *IEEE Transactions on Neural Networks*, *3*(2), 224–231.

Kim, J., Moon, Y., & Zeigler, B. P. (1995). Designing fuzzy net controllers using genetic algorithms. *IEEE Control Systems Magazine*, *15*(3), 62–72.

Kowalczuk, Z., Suchomski, P., & Bialaszewski, T. (1999). Evolutionary multi-objective pareto optimisation of diagnostic state observers. *International Journal of Applied Mathematics and Computer Science*, *9*(3), 689–709.

Koza, J. R. (1992). *Genetic programming—on the programming of computers by means of natural selection*. Cambridge, MA: The MIT Press.

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*, *1*, 121–164.

Krishnakumar, K., Swaminathan, R., & Montgomery, L. (1994). Multiple optimal solutions for structural control using genetic algorithms with niching. *Journal of Guidance and Control*, *17*(6), 1374–1377.

Kristinsson, K., & Dumont, G. A. (1992). System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(5), 1033–1046.

Langdon, W.B., & Poli, R. (1997). Fitness causes bloat. *Proceedings of the second on-line world conference on Soft Computing in Engineering Design and Manufacturing*. Cyberspace (pp. 13–22).

Lennon, W. K., & Passino, K. M. (1999). Intelligent control for brake systems. *IEEE Transactions on Control Systems Technology*, *7*(2), 188–202.

Leontaritis, I.J., & 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*, *41*(2), 303–328 and 329–344.

Linkens, D.A., & 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*, *142*(3), 161–176 and 177–185.

Linkens, D. A., & Nyongesa, H. O. (1996). Learning systems in intelligent control: An appraisal of fuzzy, neural and genetic control applications. *IEE Proceedings—Control Theory and Applications*, *143*(4), 367–386.

Luh, G. C., & Wu, C. Y. (1999). Non-linear system identification using genetic algorithms. *Proceedings of the Institution of Mechanical Engineers Part I*, *213*, 105–118.

Marcu, T. (1998). A multiobjective evolutionary approach to pattern recognition for robust diagnosis of process faults. *Proceedings of SAFEPROCESS '97: Fault detection, supervision and safety for technical processes 1997*. Hull, UK (pp. 1183–1188).

Marenbach, P., Bettenhausen, K. D., Freyer, S., Nieken, U., & 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 Part I*, *211*, 325–332.

Marrison, C. I., & Stengel, R. F. (1997). Robust control system design using random search and genetic algorithms. *IEEE Transactions on Automatic Control*, *42*(6), 835–839.

Matsuura, K., Shiba, H., Hirotsune, M., & Nunokawa, Y. (1996). Optimal control of sensory evaluation of the sake mashing process. *Journal of Process Control*, *6*(5), 323–326.

Mei, T. X., & Goodall, R. M. (2000). LQG and GA solutions for active steering of railway vehicles. *IEE Proceedings—Control Theory and Applications*, *147*(1), 111–117.

Michalewicz, Z. (1996). *Genetic algorithms+data structures=evolution programs*. Berlin: Springer.

Michalewicz, Z., & Fogel, D. B. (2000). *How to solve it: modern heuristics*. Berlin: Springer.

Miller, J. A., Potter, W. D., Gandham, R. V., & Lapena, C. N. (1993). An evaluation of local improvement operators for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(5), 1340–1351.

Moriyama, H., & Shimizu, K. (1996). On-line optimisation of culture temperature for ethanol fermentation using a genetic algorithm. *Journal of Chemical Technology and Biotechnology*, *66*(3), 217–222.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, *1*(1), 25–49.

Obayashi, S., Sasaki, D., Takeguchi, Y., & Hirose, N. (2000). Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE Transactions on Evolutionary Computation*, *4*(2), 182–187.

Oliveira, P., Sequeira, J., & Sentieiro, J. (1991). Selection of controller parameters using genetic algorithms. In S. G. Tzafestas (Ed.), *Engineering Systems with Intelligence. Concepts, Tools, and Applications* (pp. 431–438). Dordrecht: Kluer.

Onnen, C., Babuška, R., Kaymak, U., Sousa, J. M., Verbruggen, H. B., & Isermann, R. (1997). Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, *5*(10), 1363–1372.

Painton, L., & Campbell, J. (1995). Genetic algorithms in optimization of system reliability. *IEEE Transactions on Reliability*, *44*(2), 172–178.

Patton, R. J., Chen, J., & Liu, G. P. (1997). Robust fault detection of dynamical systems via genetic algorithms. *Proceedings of the Institution of Mechanical Engineers Part I*, *211*, 357–364.

Porter, B., & Jones, A. H. (1992). Genetic tuning of digital PID controllers. *Electronics Letters*, *28*(9), 843–844.

Rana, A. S., & Zalzala, A. M. S. (1997). Collision-free motion planning of multiarm robots using evolutionary algorithms. *Proceedings of the Institution of Mechanical Engineers Part I*, *211*, 373–384.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart: Frommann-Holzboog.

Ren, Y., & Bechta Dugan, J. (1998). Design of reliable systems using static and dynamic fault trees. *IEEE Transactions on Reliability*, *47*(3), 234–244.

Rodríguez-Vásquez, K., Fonseca, C.M., & Fleming P. J. (1997). Multiobjective genetic programming: A nonlinear system identification application. *Late breaking papers at the 1997 genetic programming conference*. Stanford, USA (pp. 207–212).

Schroder, P., Green, B., Grum, N., & Fleming, P. J. (2001). On-line evolution of robust control systems: An industrial active magnetic bearing application. *Control Engineering Practice*, *9*(1), 37–49.

Schubert, W. M., & Stengel, R. F. (1998). Parallel synthesis of robust control systems. *IEEE Transactions on Control Systems Technology*, *6*(6), 701–706.

Schwefel, H.P. (1965). *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik*. Diploma Thesis, Technical University of Berlin.

Sette, S., Boullart, L., & Langenhove, L. V. (1998). Using genetic algorithms to design a control strategy of an industrial process. *Control Engineering Practice*, *6*(4), 523–527.

Thompson, H. A., Chipperfield, A. J., Fleming, P. J., & Legge, C. (1999). Distributed aero-engine control systems architecture selection using multi-objective optimisation. *Control Engineering Practice*, *7*(5), 655–664.

Tzes, A., Peng, P. Y., & Guthy, J. (1998). Genetic-based fuzzy clustering for DC-motor friction identification and compensation. *IEEE Transactions on Control Systems Technology*, *6*(4), 462–472.

Varšek, A., Urbančič, T., & Fillipič, B. (1993). Genetic algorithms in controller design and tuning. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(5), 1330–1339.

Veldhuizen, D. A. V., & Lamont, G. B. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, *8*(2), 125–147.

Vlachos, C., Williams, D., & Gomm, J. B. (1999). Genetic approach to decentralised PI controller tuning for multivariable processes. *IEE Proceedings—Control Theory and Applications*, *146*(1), 58–64.

Wang, P., & 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)*, *3*, 1224–1229.

Wang, Q., & Stengel, R. F. (2000). Robust nonlinear control of a hypersonic aircraft. *Journal of Guidance, Control, and Dynamics*, *23*(4), 3341–3346.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Zakian, V., & Al-Naib, U. (1973). Design of dynamical control systems by the method of inequalities. *Proceedings of the IEE*, *120*, 1421–1427.

Zhang, H., Lennox, B., Goulding, P. R., & 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*, *9*, 552–557.