EVOLUTIONARY ALGORITHMS IN CONTROL SYSTEM ENGINEERING

Daniel R. Lewin

PSE Research Group, Wolfson Department of Chemical Engineering,
Technion–IIT, Haifa 32000, Israel

Abstract: This paper introduces the session describing the state-of-the-art in the application of evolutionary computation in control system engineering. Evolutionary methods such as genetic algorithms (GAs) and genetic programming (GP) are particularly suitable in problems for which conventional optimizers are inefficient or inappropriate, rather than simply as an alternative to conventional optimization. This session will be of interest to the control engineering community as a whole, and will provide an educational background to those not familiar with these methods as well as presenting new results and applications.

Keywords: Genetic algorithms; genetic programming; model predictive control.

1. INTRODUCTION

Completeness and harmony in nature are largely the result of evolutionary forces that have adapted species to their surroundings and to each other. Examining natural phenomena, one can appreciate the potential of nature's ready-made solutions, which are often more efficient than man-made ones. For example, consider spider silk, which is more elastic than nylon and stronger than steel, and witness the abilities of the spider, essentially blind with a limited nervous system, to use six variants of silk to build a robust, complex structure in an unpredictable environment (Conniff, 2001).

With nature as a motivator, recent decades have seen increasing attempts to mimic natural evolution using computers (Fogel et al., 1965; Holland, 1975; Koza, 1992). These efforts are stimulated by Darwin's notion of "the survival of the fittest," are generally referred to as *evolutionary computation*, and have been applied successfully to solve particularly complex problems. In evolution, the problem each species faces is one of searching for beneficial adaptations to a changing environment, with the "lessons" learned captured in the chromosomes of its population. Furthermore, evolutionary methods keep track of *populations* of potential solutions, and are thus less sensitive to arbitrary initial guesses of the solution than classical optimization methods, which often rely on local gradient search.

This paper presents a review of the state-of-the-art in the two most prevalent evolutionary computation methods that are applied to control systems engineering: genetic algorithms and genetic programming.

2. GENETIC ALGORITHMS

The most commonly-used evolutionary computation algorithm is the *genetic algorithm* (GA). In most implementations, these algorithms manipulate binary encodings of the decision variables to be optimized, which are concatenated into so-called *chromosomes*. Mimicking nature, the algorithm starts its search from an initial population of solutions, usually generated randomly, or occasionally, based on problem-specific knowledge. The performance of each individual is evaluated using a fitness function that gauges its performance, with the most successful (efficient) chromosomes having a higher probability to reproduce. In synthetic evolution, reproduction is similar to that in biological reproduction, which by mimicking natural operators like *crossover* and *mutation* creates a generation of offspring solutions. *Crossover* generates new features in the solution

space by combining genetic information, while *mutation* does this by adding random perturbations. The subsequent generations are subjected to these evolutionary operators, thus producing generation after generation of offspring solutions. Since the more appropriate solutions are given higher probabilities to reproduce, one would expect a growing improvement of the solutions over generations.

It has been shown that genetic algorithms are efficient and appropriate optimization methods for control system design (see the excellent survey by Fleming and Purshouse, 2002), for both feedback controllers (e.g. Fonseca and Fleming, 1993; Lewin, 1994; Tang et al, 1996; Fonseca and Fleming, 1998; Lewin and Parag, 2003) and feedforward controllers (Lewin, 1996). All these studies have involved the optimization of the parameters of a control system of fixed structure in order to achieve robust stability and specified performance. The multi-objective approach (MOGA), first suggested by Fonseca and Fleming (1993), permits the modification of the objective as performance is gauged in the course of optimization. This methodology allows multiple, often non-commensurate, performance objectives to be handled separately, and allows goals and priorities for these to be updated by the decision-maker during the course of the optimisation as the effect of trade-offs between them are discovered. MOGAs have been successfully applied in the design of several complex control problems, such as a gas turbine aero-engine, relying on a classical control framework (Chipperfield and Fleming, 1996), as well as in several contributions in this session (Ferreira et al. (2005); Molina-Cristóbal et al, 2005; Stirrup and Chipperfield, 2005). It is noted that the MOGA approach is a generalization of the trade-off line approach advocated by Lewin (1994), in the automated design of MIMO systems. Exploiting the population-based nature of GAs, Lewin (1996) has shown that statistical hypothesis testing can be applied to a population of successful solutions in order to eliminate controller parameters, which have little impact on the optimal solution. Lewin and Parag (2003) presented a constrained genetic algorithm (CGA), in which the desired trade-off between robustness and performance is incorporated into the control design procedure for uncertain processes. The following section provides a demonstration of the capabilities of a GA to solve a nonlinear control problem, featuring a discontinuous search space.

3. AN EXAMPLE GA APPLICATION

3.1 Problem definition.

Consider the neutralization process in Figure 1, in which an acidic waste stream of HCl, F_1, is to be neutralized using a controlled stream of NaOH, F_2. The control objectives are to maintain the effluent stream, F, at a pH of 7, despite possible upsets in the flowrate and HCl concentration in F_1. This is a difficult control design problem because the neutralization curve is highly nonlinear, meaning that the effective gain of the process, the local gradient of the curve in Figure 2, rises by several orders of magnitude as the setpoint is approached.

Following Shinskey (1979), a PI controller with reset time τ_I, and with a three-piece nonlinear gain, K_c, is adopted to overcome this difficulty, with a low gain used when the setpoint tracking error is inside a prespecified range, and a high gain value used when outside the range:

$$K_c = \begin{cases} K_{high}, & |e| \geq e_{band} \\ K_{low}, & |e| < e_{band} \end{cases}, e = pH - pH_{setpoint} \qquad (1)$$

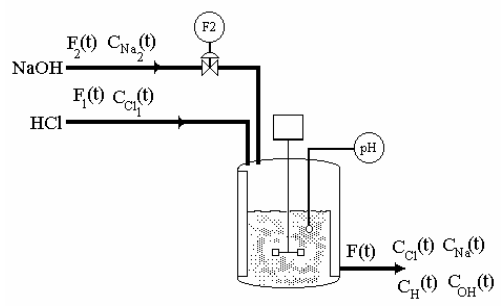

Fig. 1. Flowsheet of the neutralization process.

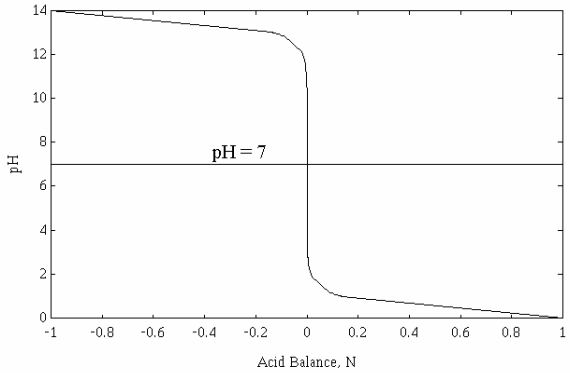

Fig. 2. Neutralization curve.

3.2 Solution using GA.

The proposed control system, using the model of Kulkarni et al (1991) and simulated in SIMULINK®, is shown in Figure 3, noting that the disturbance to be suppressed is a step change in the flow rate of the acidic stream. To avoid ringing phenomena when switching from high to low gain values, a low-pass filter, with filter time constant τ_F, is inserted after the logic circuit that switches between the values. Thus, five parameters need to be optimized: K_{high}, K_{low}, e_{band}, τ_I, and τ_F. Considering the discontinuous nature of the control design, and the significant nonlinearities in the system, this is a problem that merits an evolutionary approach.

Figure 4 shows typical performance of the GA in the solution of this problem. The fitness function presented to the GA is the ITAE computed by simulation using the five tunable parameters for which optimal values are sought. Typical computation time for a population of 30 solutions propagated over 30 generations is about seven minutes on a 1.6 GHz Pentium 5 (faster times are possible if the graphical

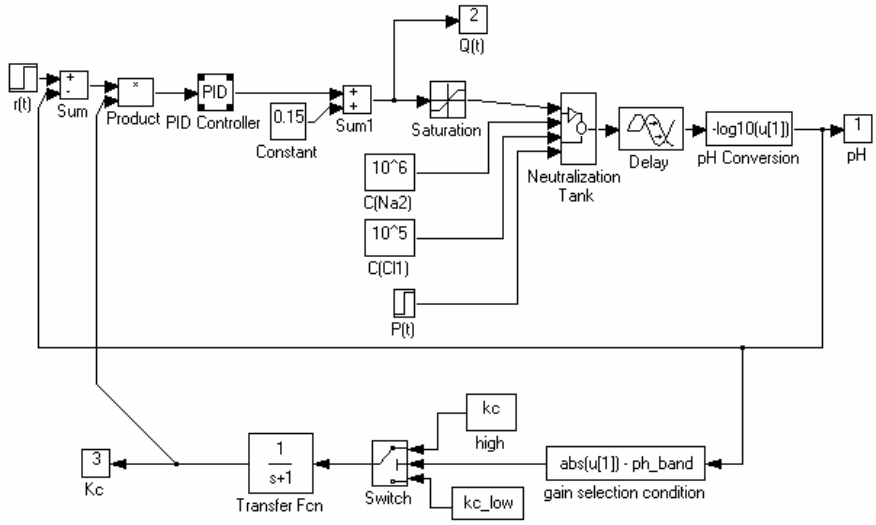

Fig. 3. SIMULINK® model of the neutralization process, controlled by a three-piece nonlinear controller.

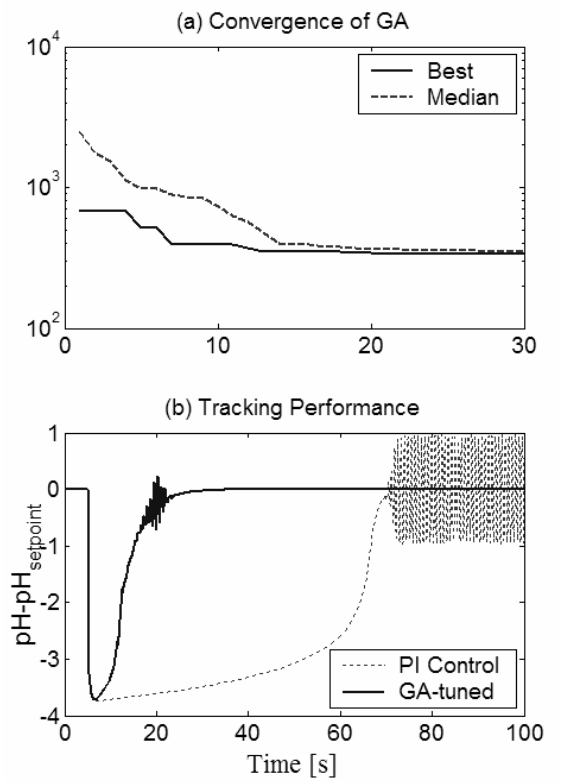

Fig. 4. Performance of the GA: (a) Convergence properties showing evolution of best and median solution ITAE values; (b) Response of best GA-evolved solution (solid lines) compared to linear PI (dotted lines).

output of the MATLAB®-based code is suppressed). Figure 4(b) shows that the GA-tuned nonlinear controller significantly outperforms a fixed-gain linear PI controller, whose gain needs to be set to be large enough to be able to reject the step disturbance. The relatively large gain required subsequently causes the oscillatory response in the vicinity of the setpoint, because of the exceedingly large process gain in the vicinity of the setpoint (pH = 7).

4. GENETIC PROGRAMMING

While the efficiency of genetic algorithms in a number of fields has been demonstrated, they are unsuitable for generating empirical model structures, since they manipulate populations of solutions of **fixed-length** chromosomes, while the optimal complexity of empirical models is unknown in advance. Because of this perceived need for more intelligent construction of empirical models, a more recent family of evolutionary computation methods has emerged, based on established GA ideas. These new algorithms, referred to as *genetic programming* (GP), rely on tree-like building blocks and therefore support populations of model structures of varying length and complexity. Activity in Genetic Programming was introduced by Koza (1992), who demonstrated their applications in robotics, games, control, and symbolic regression.

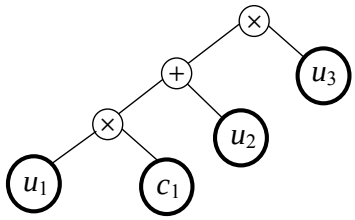

Fig. 5. Tree structure for the model: $(c_1 \times u_1 + u_2) \times u_3$. The multiplication functional is positioned at the root of the tree, and u_3 and $c_1 u_1 + u_2$ are its branches.

One of the important applications of genetic programming is in generating input-output empirical models in systems engineering applications. The class of empirical models can be divided into two broad categories: (a) models with predefined structure (either linear or nonlinear), and whose parameters are determined to maximize the capacity to predict process data; and (b) black-box models with undetermined structure. An example of the first category would be a linear model relating a dependent variable, y, to a set of n independent variables, u_i:

$$y = \sum_{i=1}^{n} a_i u_i \,, \qquad (2)$$

where the coefficients a_i are determined to maximize the predictive power of the model. An example of a black-box model would be an artificial neural network (ANN), in which the number and identity of the relevant inputs and the number of layers are the only attributes of the structure that are determined by the user. The disadvantage of the first alternative is that the user must specify the structure of the model in advance, which is in general difficult to do. In contrast, the main disadvantage of the neural network approach is that no formal equation is obtained, and thus, the resulting model is difficult to analyze. Consequently, great care must be taken with the ANN approach to prevent over-fitting. Genetic programming, being an evolutionary method for automatically generating nonlinear input-output models, overcomes both of the disadvantages mentioned above, since structured models are obtained, whose complexity is optimized.

When applying genetic programming to automatically generate nonlinear MISO (multiple input, single output) models, the probability of a given model surviving into the next generation is proportional to how well it predicts the output data. Components of successful models are continuously recombined with those of others to form new models. The GP optimizes the model structure, with a lower level nonlinear least-squares algorithm harnessed to perform the associated parameter estimation.

Several publications describe the usage of GP for nonlinear process modeling. South et al. (1996) introduce a Genetic Programming called GAFFER (Genetic algorithm for finding existing relationships), an algorithm that combines both logical and mathematical operators. They demonstrate the use of the code to identify the parameters of a first order discrete model, with a known structure – an application typical of what could easily be solved with a GA – and in finding the structure and parameters of a simple discrete model. McKay et al. (1997) use a GP for inferential estimation in a vacuum distillation column and in reaction system modeling. Willis el al. (1997), and Hinchliffe and Willis (2003) apply GP to both steady-state and dynamic process modeling, and demonstrate their algorithm on the steady state modeling of a binary distillation column, and on discrete dynamic modeling of a twin screw cooking extruder. Gray et al. (1996) implement a GP for dynamic modeling, in which a GP generates continuous discrete models, by combining MATLAB-SIMULINK blocks with sets of equations.

Grosman and Lewin (2004) present an improved GP with the capacity to generate compact nonlinear models that accurately predict the input-output system behaviour without requiring the user to specify the model complexity in advance; instead, the required model complexity is adapted to the degree required to appropriately model the data, eliminating the "bloat" phenomena indicated as one of the main disadvantages of the GP approach by Fleming and Purshouse (2002). Grosman and Lewin (2002) use their GP to automate the design of nonlinear MPC (NMPC). Lachman-Shalem et al (2002) demonstrate the exploitation of Grosman's NMPC methodology for the control of a complete simulated photolithography cluster, which has now been implemented on an industrial IC fabrication facility (Grosman et al, 2005; Lewin et al, 2005). In this session, Grosman and Lewin (2005) present a method for automatically deriving Lyapunov functions for nonlinear systems stability analysis using the adaptive GP. A demonstration of the usage of the adaptive GP code (Grosman and Lewin, 2004) to provide a dynamic model for model predictive control follows next.

5. AN EXAMPLE GP APPLICATION

5.1 Problem definition.

Consider the relatively simple illustrative example involving a stirred tank fed by two streams a simulated cylindrical mixing tank, consisting of two feed flows: one of fresh water and the other of saturated salt water. The objective is to ensure the fluid level in the tank and the effluent salt concentration are maintained at the desired set points. The total and salt mass balances are:

$$A\frac{dH}{dt} = Q_w + Q_s - k_0 \cdot \sqrt{H} \qquad (3)$$

$$A\frac{d\left(C \cdot H\right)}{dt} = Q_w \cdot C_w + Q_s \cdot C_s - k_0\sqrt{H} \cdot C \qquad (4)$$

In the above equations, H is the fluid level in the tank, C is the tank salt concentration, A is the cross-sectional area of the tank (=1), Q_w is the sweet water flow; Q_s is the salt water flow, C_w is the sweet water salt concentration, (=0); C_s is the salt water concentration (=1); k_0 is the valve constant (=1). Since Q_w and Q_s are normalized to lie between zero and one, the normal operating ranges of the concentration and the height are also in this range. Thus, a level value in excess of unity implies overflow of the tank, and indicates catastrophic loss of control.

5.2 Creation of state-space model using GP

Discrete input-output models are generated to allow the prediction of level and concentration trajectories using the GP. Delayed outputs ($k-1$, $k-2$, $k-3$) and inputs (k, $k-1$, $k-2$, $k-3$) are presented to the GP for predicting $y(k)$. Pseudo-random sequences of steps in the two inputs are used to excite the process, with two sets generated – one for modeling and one for prediction. The sampling time is selected to be 0.15 time units, to capture desired features of the process response. The use of trajectory matching for both modeling and prediction means that our approach is relatively insensitive to the sampling rate selected, which is one of its advantages. The models that score the highest are:

$$\begin{aligned} C\left(k\right) = {} & 0.818 \cdot C\left(k-1\right) + 0.134 \cdot Q_s\left(k-1\right) \\ & - 0.078 \cdot Q_w\left(k-2\right) + 0.68 \end{aligned} \qquad (5)$$

$$H(k) = 0.890 \cdot H(k-1) + 0.170 \cdot Q_s(k) + 0.167 \cdot Q_w(k) - 0.061 \quad (6)$$

It is noted that both models are linear, which is not surprising, noting that the actual process is itself almost linear. This suggests that a linear MPC strategy would be expected to do well on this process, which has been confirmed by simulation.

5.3 Nonlinear MPC

Model predictive control is a multivariable control strategy in which a process model is used to predict the effect of previous control actions and enable the optimization of future control moves, to ensure process output variables are maintained on target. The nonlinear MPC controller presented by Grosman and Lewin (2002) is similar to that outlined by Henson (1998), with the main difference being the nonlinear GP model used to predict the future outputs, and consists of several components:

a. A MISO input-output model for each process output, created by the GP, which takes the general form:

$$\hat{y}(k) = f\left(\underline{\hat{y}}(k-1|k), \underline{u}(k-1|k) \right) \quad (7)$$

where $\hat{y}(k)$ is the predicted model output at instant k, $\underline{\hat{y}}(k-1|k)$ is a vector of the previous P outputs, computed based on information available until instant k, and $\underline{u}(k-1|k)$ is a vector of P inputs at instant k. The approach also handles nonlinear state-space models.

b. Selected values for P, the prediction horizon, that controls the predictive range of the model, and M, the control horizon, that establishes the number of future moves to be optimized, noting that $M \leq P$. In this application, these parameters are selected as $P = 10$ and $M = 5$. As in all MPC strategies, only the first of the computed controller moves is implemented at each sample interval.

c. A matrix of historical data used by the model to predict the influence of previous outputs and inputs on the future outputs. The matrix has P rows and a number of columns equal to the total number of shifted inputs and outputs, and is refreshed every sample instant.

d. A constrained quadratic objective function to be minimized, whose arguments are calculated on the basis of predicted values of outputs generated using the nonlinear GP models, resulting in a sequence of M optimal future inputs.

5.4 Comparison of the GP-NMPC approach with PI Control

A commonly used approach to multivariable control system implementation is decentralized PI control, often selected because it is easily implemented. Thus, the performance of the GP-NMPC approach is compared here to that of a decentralized PI control system, with pairings: $H - Q_W$, $C - Q_S$. Here, we report two representative tests that indicate the regulatory and servo performance of the controllers.

Regulatory performance: A pulse of 40% in the influent salt-water flow is invoked, starting from the steady state [$H(0)$, $C(0)$] = [0.5, 0.5]. The GP-NMPC is set up with a prediction horizon of 10 and a control horizon of 5, with all weights in the quadratic function set to unity, noting that inputs and outputs are scaled by definition. The response of the GP-NMPC and the PI control configurations are shown in Figure 6. It is noted that GP-NMPC provides almost perfect disturbance rejection, whereas the response using the decentralized PI controller is oscillatory, indicative of the strong coupling in the system. Detuning the PI controllers reduces the intensity of the oscillations, but at the price of increased settling time.

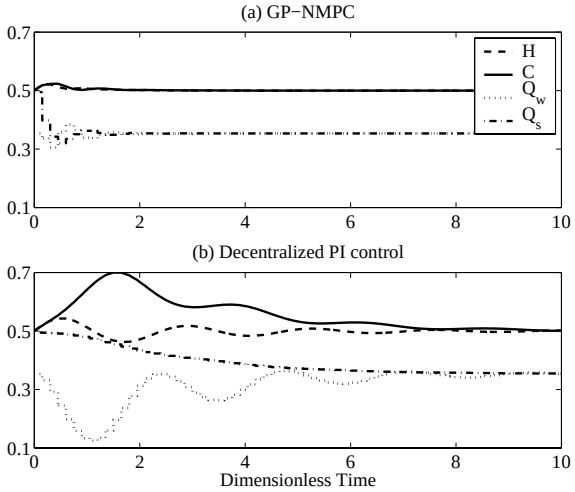

Fig. 6. Regulatory performance of the NMPC and PI controllers: (a) GA-NMPC; (b) Decentralized PI Control.

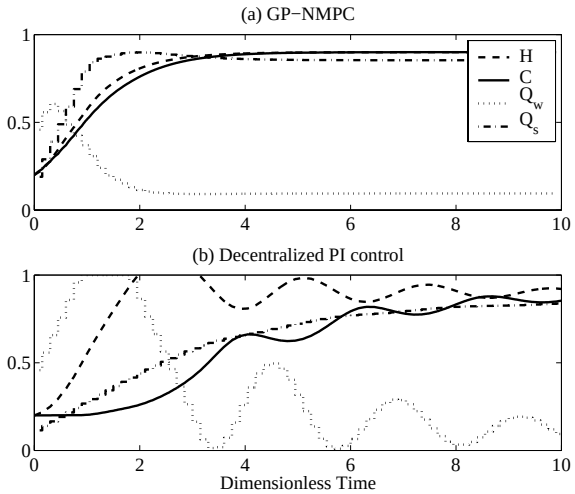

Fig. 7. Servo performance of the NMPC and PI controllers: (a) GA-NMPC; (b) Decentralized PI Control.

Servo performance: A setpoint change in both outputs is commanded, from the steady state [$H(0)$, $C(0)$] = [0.2, 0.2] to values at [0.9, 0.9], noting that these are close to the upper constraints of the process. The GP-NMPC is set up as in the regulatory test,

with its response, together with that of the PI control configuration, shown in Figure 7. The GP-NMPC provides rapid acquisition of the targets, while satisfying the process hard constraints. In contrast, the decentralized PI controller, again strongly oscillatory, violates the upper level constraint, leading to overflow of the tank in the range $2 < t < 3$ (in dimensionless time units).

6. CONCLUSIONS

GAs and GPs mimic the natural evolutionary process through representation of engineering design parameters, in the case of GAs, and of model structures, in the case of GPs, as genes, and permitting them to evolve towards optimal solutions. Through the mechanisms of evolution and genetics, biological systems optimize and adapt their development. Adopting these same mechanisms, GAs and GPs exhibit great potential for the solution of complex control engineering design problems working within the constraints of their environment. Using these algorithms, a wider range of control systems design problems can be practically addressed. This paper, together with the other five contributions in the invited session *Evolutionary Computation in Control Systems Engineering,* provide an update on the state-of-the-art in evolutionary computation as applied to control system design and optimization.

REFERENCES

Chipperfield, A. and P. Fleming (1996). Multiobjective GAs Turbine Engine Controller Design using Genetic Algorithms. *IEEE Transactions Industrial Electronics,* **43**(5) 583.

Conniff, R. (2001). "Deadly Silk – Spiderwebs, *National Geographic*, **200**(2), 30.

Ferreira, P.M., A. E. Ruano, and C. M. Fonseca. (2005). Evolutionary Multiobjective Design of Radial Basis Function Networks for Greenhouse Environmental Control, *Proc. of the 16th IFAC World Congress*, Paper 4459.

Fleming, P. J. and R. C. Purshouse (2002). Evolutionary Algorithms in Control Systems Engineering: A Survey. *Control Engineering Practice*, **10**(11), 1223.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1965), in *Artificial Intelligence through a Simulation of Evolution*, Maxfield, M., Callahan, A. and L. J. Fogel (eds.), 131-155, Spartan Books, Washington.

Fonseca, C. M. and P. J. Fleming (1993). Multiobjective Genetic Algorithms, *IEE Colloquium on `Genetic Algorithms for Control Systems Engineering',* 6/1-5.

Fonseca, C. M. and P. J. Fleming (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part 1: A Unified Formulation. *IEEE Trans. Sys., Man. & Cyber.*, **28**, 26.

Goldberg D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Gray, G. J., D. J. Murray-Smith, Y. Li and K. C. Sharman (1996). Nonlinear Model Structure Identification using Genetic Programming and a Block Diagram Oriented Simulation Tool, *Electronic Letters,* **32**, 1422.

Grosman, B. S. Lachman-Shalem, R. Swissa and D. R. Lewin (2005). Yield Enhancement in Photolithography through Model-based Process Control: Average Mode Control, *IEEE Trans. of Semiconductor Manufacturing*, **18**(1) 86.

Grosman, B. and D. R. Lewin (2002). Automated Nonlinear Model Predictive Control using Genetic Programming, *Comput. Chem. Eng*, **26**(4-5), 631.

Grosman, B. and D. R. Lewin. (2004). Adaptive Genetic Programming for Steady-state Process Modeling, *Comput. Chem. Eng.,* **28**(12), 2779.

Grosman, B. and D. R. Lewin (2005). Automatic Generation of Lyapunov Functions using Genetic Programming, *Proc. of the 16th IFAC World Congress*, Paper 2843.

Henson, M. A. (1998). Nonlinear Model Predictive Control: Current Status and Future Directions, *Comput. Chem. Engng*., **23**(2), 187.

Hinchliffe, M. P. and M. J. Willis (2003). Dynamic Systems Modelling using Genetic Programming. *Comput. Chem. Engng.*, **27**(12), 1841.

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

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Kulkarni, B. D. and S. S. TambeNeelkant ,V. Shukla and P. B. Deshpande (1991). Nonlinear pH Control, *Chem. Eng. Sci*., **46**(4), 995.

Lachman-Shalem, S., B. Grosman, and D. R. Lewin (2002). Nonlinear Modeling and Multivariable Control of Photolithography, *IEEE Trans. of Semiconductor Manufacturing*, **15**(3), 310.

Lewin, D. R. (1994). A Genetic Algorithm for MIMO Feedback Control System Design, *IFAC Symposium on Advanced Control of Chemical Processes* (ADCHEM'94), Kyoto, Japan.

Lewin, D. R. (1996). Multivariable Feedforward Control Design Using Disturbance Cost Maps and a Genetic Algorithm, *Comput. Chem. Eng*., **20**(12), 1477.

Lewin, D. R. and Parag, A. (2003). A Constrained Genetic Algorithm for Decentralized Control System Structure Selection and Optimization, *Automatica*, **39**, 1801.

Lewin, D. R., S. Lachman-Shalem, and B. Grosman (2005). More PSE Applications in IC Manufacturing, *Proc. of the 16th IFAC World Congress*.

McKay, B., M. J. Willis and G. W. Barton (1997). Steady-state Modeling of Chemical Processes using Genetic Programming, *Comput. Chem. Engng.*, **21**(9), 981-996.

Molina-Cristóbal, A., I. A. Griffin, P. J. Fleming, and D. H. Owens (2005). Multiobjective Control Design: Optimizing Controller Structure with Genetic Algorithms, *Proc. of the 16th IFAC World Congress*, Paper 3860.

Shinskey, G. (1979). *Process Control Systems*, McGraw-Hill, New York.

Stirrup, R. and A. J. Chipperfield (2005). Improved MOGA-Tuning and Visualization for a Hybrid Control System, *Proc. of the 16th IFAC World Congress*, Paper 2906.

Tang, K. S., K. F. Man and D.-W. Gu (1996). Structured Genetic Algorithm for Robust H^{∞} Control Systems Design. *IEEE Transactions Industrial Electronics*, **43**(5) 575.

Willis, M. J., H. Hiden, M. Hinchliffe, B. McKay and G. W. Barton (1997). Systems Modeling using Genetic Programming, *Comp. Chem. Engng.*, **21**(S), S1161.