Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

SEQUENTIAL METAMODELLING WITH GENETIC PROGRAMMING AND PARTICLE SWARMS

Birkan Can
Cathal Heavey

Enterprise Research Centre
University of Limerick
Limerick, IRELAND

ABSTRACT

This article presents an application of two main component methodologies of evolutionary algorithms in simulation-based metamodelling. We present an evolutionary framework for constructing analytical metamodels and apply it to simulations of manufacturing lines with buffer allocation problem. In this framework, a particle swarm algorithm is integrated to genetic programming to perform symbolic regression of the problem. The sampling data is sequentially generated by the particle swarm algorithm, while genetic programming evolves symbolic functions of the domain. The results are promising in terms of efficiency in design of experiments and accuracy in global metamodelling.

1 INTRODUCTION

Discrete-event simulation (DES) can be regarded as an evaluative tool to support decision making in system studies. The efficiency of models in understanding system behaviour and measuring the performance against different system configurations (or designs) is a crucial aspect in analysis and optimisation of systems. Generally, a large system implies long model execution times. Furthermore, most real-world systems exhibits stochastic elements such as variability of demand, failures in processes, etc.. Incorporation of such aspects to models necessitates special statistical approaches for credible performance measurements. However, this will require multiple or long simulation runs. These can render the potential of simulation prohibitive in practice. In such circumstances approximating simulation models by metamodelling can present an appropriate trade-off between accuracy and efficiency.

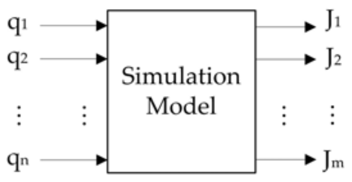

Figure 1: An illustration of mapping between simulation input (decisions) and output (performance)

In most system studies, it is a common practice to consider a DES model as a mapping between decision variables and system performance as shown in Figure 1. Hence, metamodelling can construct a black-box or an analytical model which can represent the functionality of the actual DES model itself. An exploratory review on the use of metamodelling can be found in Wang and Shan (2007).

A metamodelling study, in general, consists of sampling of decision space, model construction and its validation in order to obtain a substitute for the computationally expensive DES models. These steps span a variety of different techniques. Altiparmak et al. (2007) present a comparative study in metamodelling with regression and artificial neural networks on assembly lines. Beers and Kleijnen (2004) exemplify the application of a promising technique, Kriging. Jin et al. (2001) present a comparative analysis of different model construction techniques on a range of test problems. The techniques do not present much diversity in terms of success characteristics on problems of different sizes and response types.

Kleijnen et al. (2005) provide an extensive review on design of experiments, i.e. sampling, regarding simulation. The article points out that the selection of an experimental design procedure depends on the model type assumed to fit to sampled (or training) data. These outcomes from the comparative studies motivate the need for interrelated approaches for both design of experiments and model construction phases.

In this article, we will try to address this observation by integration of two mainstream evolutionary algorithms (EAs). In this framework, we will exploit genetic programming (GP) for constructing analytical approximations to DES models and particle swarm optimisation (PSO) to assist the process by generation of new sample designs when required by GP. The article will first introduce these techniques in the context of metamodelling in Sections 2.1 and 2.2. Subsequently, an overview of the framework will be presented, followed by an example application on DES models of production lines with buffer allocation problems (Section 3.2). Discussion and notes for future will finalise the article.

2 METAMODELLING

As mentioned earlier in Section 1, a DES model can be considered as a black-box transformation of decision variables into performance metrics (see Figure 1). This motivates metamodelling studies focusing on obtaining a mapping function between decision variables and system performance as in (1).

$$f(Q) \approx J(Q) + \varepsilon \tag{1}$$

In this context, one can try to obtain an approximate function, $f(Q)$ of the performance measure, $J(Q)$, dependent on a set of decision variables, Q, while minimising an error measure, ε (1). In this work, this task will be performed in an evolutionary framework in order to obtain plausible metamodels with fewer set of experimental designs. A step-wise review of the content regarding design of experiments and metamodel construction phases will be presented in the following subsections.

2.1 Design of Experiments

Design of experiments (DOE), or sampling, involves extracting a number of system designs from the problem decision space. Clearly, it is an important aspect for any metamodelling study since the sample points identify the system behaviour. Therefore, DOE has received a significant attention yielding a variety of systematic methods to assist design and analysis of systems.

The literature presents numerous applications of single stage space filling methods (Latin Hypercube Sampling, Uniform Designs, etc..) which adjust the spread of design points with respect to an optimality criteria. However, considering only the spread of possible sample points may undermine accuracy of metamodelling particularly when the system exhibits unknown nonlinear behaviour (Kleijnen et al. 2005). Moreover, a discrete design space with combinatorial relationships among the decision variables can further complicate the experimental designs. In particular, existence of problem constraints and a large decision space will prohibit the success in applications of these techniques. In order to respond to these shortcomings, sequential sampling procedures have become attractive alternatives.

Sequential sampling can be regarded as retrieval of new design points from the decision space when required during the metamodelling procedure. There have been variety of occasions where sequential sampling is applied to perform metamodelling (Kleijnen and Beers 2003, Li et al. 2006). Hendrickx et al. (2006) demonstrate an application of artificial neural networks with sequential sampling driven by the accuracy of the metamodels. Similarly, Balasko et al. (2006) considers sampling as a nonlinear optimisation problem. They integrate Evolution Strategies (ES) to metamodelling procedure to generate optimal designs. In this manner, ES explores the decision space and locates design points which maximise a fitness function obtained from Fisher information matrix. The approaches considering the optimality of training points and availability of numerous techniques motivate the exploitation of particle swarm optimisation (PSO) to assist metamodelling in this article.

Particle swarm optimisation is an evolutionary algorithm which emulates social influence to maintain a cognitive optimisation procedure. PSO was originally developed for solving continuous nonlinear optimisation problems. It consists of simple concepts and mathematical operations with little memory requirements. Therefore, it is fast and appealing in use for many optimisation problems (Kennedy and Eberhart 1995). It has also been shown to be efficient in discrete problems (Liu et al. 2008). Banks et al. (2007) present a review on the background of PSO, Alkhamis and Ahmed (2005) exemplify an application on simulation-based optimisation, Kashan and Karimi (2009) use PSO for a job-shop scheduling problem.

In PSO, each solution is defined as a particle and kept in a social network called swarm. A solution, x_i corresponds to a position, $x_{i=}(x_{i1}, x_{i2}, \ldots, x_{in})$ in a n-dimensional space. Each individual particle keeps track of the position, $P_{i=}(p_{i1}, p_{i2}, \ldots, p_{in})$, where it has reached its highest quality with respect to the function to be optimised. Similarly, P_g, keeps the global best position of all particles in the swarm. These particles iteratively move through the search space with a velocity, $V_{i=}(v_{i1}, v_{i2}, ..., v_{in})$. The particle's velocity is dependent on its P_i, P_g of the swarm, and the preceding velocity of the particle by inertia weight, ω. The update mechanism of these components are given in (2); where t is the iteration number and $r_i(i = p, g)$'s are random numbers from the interval, $[0, 1]$. The parameters; χ and c_p, c_g's are constants.

$$\begin{aligned} V_i^{t+1} &= \omega V_i^t + c_p r_p (P_i - X_i) + c_g r_g (P_g - X_i) \\ X_i^{t+1} &= X_i^t + \chi V_i^{t+1} \end{aligned} \tag{2}$$

2.2 Model Construction and Validation

In this part, we will introduce symbolic regression with genetic programming for simulation-based metamodelling. Genetic Programming (GP) is an EA which can evolve programs (Koza 1992). These programs can be grammar rules, mathematical functions, etc.. This ability allows GP to construct analytical metamodels via symbolic regression.

In a metamodelling study, sampling is followed by choosing a model, $f(x)$, and discovering a final metamodel by minimising the deviation of the assumed model iteratively via regression analysis. To illustrate, in polynomial regression a multi-variable polynomial (3) can be chosen to fit as a metamodel. The coefficients are discovered by the minimisation of residual errors against the training data through regression analysis during metamodelling.

$$f(x,y) = a_0 + a_1 x + b_1 y + a_2 x^2 + b_2 y^2 + \dots \tag{3}$$

Similarly, GP has the ability to construct analytical functions based on its functional primitives and a terminal set. Functional primitives are simply mathematical expressions; e.g. $sine, exp, sqrt$, to capture the nonlinearities of the response surface, whereas the terminal set contains the decision variables of the problem and constants. GP evolves its solutions over these sets by combination of individual elements into a meaningful functions. Therefore, it does not require a presumption about the response model. Equation 4 exemplifies such a function for a two dimensional problem.

$$f(x,y) = \sin(\sqrt{x^3}) + \tan(\exp(y)) \tag{4}$$

The symbolic regression procedure starts with the initialisation of GP by generating random functions of pre-defined size in tree form (Figure 2). Sizes of these initial functions are generally small and limited to identify the promising building blocks of functions for further stages. Each individual function needs to be evaluated against the training data to identify how fit the metamodels are in representing the sampling data.

Equation 5 exemplifies an error function corresponding to the root mean squared error (RMSE) of an individual, $f(Q)$, against the actual simulation performance, $J(Q)$ for every design, i, in the training data. We have used $1/(1+RMSE)$ as the fitness criteria for generated metamodels. Maximisation of this fitness by means of evolutionary paradigms can lead to high quality metamodels in an optimisation context.

$$RMSE = \sqrt{1/n \sum_{\forall i} [f(Q) - J(Q)]^2} \tag{5}$$

Following the initialisation process, the individuals go through a selection process to breed the next generation. Two individuals from the current population are selected to form a new individual via crossover mechanism (Figure 2). Similarly, a mutation operator can modify individual solutions to enable local search within the solution definition. Each new solution is submitted to form the new population of the next generation. Iterative application of this concept can lead to the discovery of better analytical approximations to the simulation model.

The advantages of GP in metamodelling stem from the ability of generalisation and capturing the nonlinearities in response through functional primitives. Since the type and structure of the functions are adaptively created by means of evolution, the search process works simultaneously on the model structure and the quality of fit. Therefore, SR can be particularly useful when the structure of a response surface is unpredictable(Can and Heavey 2009). Additionally, since GP evolves analytical functions of types in functional primitives, interpretability of models and identification of key variables become possible (Murphy and Ryan 2008).

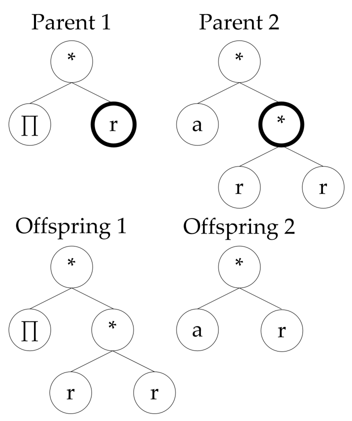

Figure 2: Tree representation of functions. The 'Offspring 1' defines the area of a circle, πr^2. A swap type node exchange crossover is illustrated with exchange nodes in bold circles.

3 EXPERIMENTS

In this section, we will perform an empirical analysis of the proposed framework (Section 3.1) on a DES problem of a manufacturing plant (Section 3.2. The results (Section 3.3) will finalise the section.

3.1 Proposed Framework

The proposed framework comprises three main components; DES, GP and PSO. We have used Plant Simulation 8.2 in order to model the BAP and the algorithms (PSO and GP) are implemented in VS .Net C#. Plant Simulation facilitates a remote control interface which allows the communication between the algorithms and simulation model through COM objects. Therefore, the framework remains quite efficient since it will not require any database in data exchange. GP constructs metamodels over the training data and requests further sampling from the PSO algorithm when needed. PSO explores the decision space with respect to an optimality criteria, excluding the design points already in the training or testing set. The fitness of a new sample is assessed with respect to information content. The PSO algorithm searches the decision space to maximize average mean absolute error, which is the error of a sample over the existing metamodels. The larger the AMAE is, higher the deviation from the metamodels. When a satisfactory solution is obtained, the simulation model is set up with respect to the configuration required by the solution. When a simulation is complete, Plant Simulation can signal the results to PSO algorithm. Subsequently, PSO submits the new design to the training data and triggers the GP. This procedure continues iteratively until a stopping criterion is satisfied. In our design, we have used two stopping criteria in parallel: (i) metamodel evaluations by GP, (ii) maximum validity of the metamodels.

The metamodelling procedure starts with a small number of samples points (4 to 8). Hence, it is possible to observe a phenomenon, called 'overfitting' at the early stages of symbolic regression. Overfitting implies an over representation of the training data, thus loss of generalisation ability. Therefore, GP needs to be controlled against over representation of training data and should implement a control mechanism to signal the need for additional sampling. During a GP run, the evolving metamodels has potential to grow in size as a result of variation operators in parallel with increasing quality of fit. Hence, one way of preventing overfitting is applying parsimony pressure which utilises a penalty functions to control the solution complexity (Poli and McPhee 2008). We implemented the hereditary repulsion algorithm provided in Murphy and Ryan (2008) due to its effectiveness. In addition, another implication of the loss of generality can be be the need for additional sampling. Observation of a change in solution size and quality may help capturing its start. Prechelt (1998) quantifies this procedure within a cross-validation framework. The idea is later investigated by Foreman and Evett (2005) for recognition of overfitting in GP with canary functions.

In the following subsections, we will first summarise the problem we studied, and then present some empirical results.

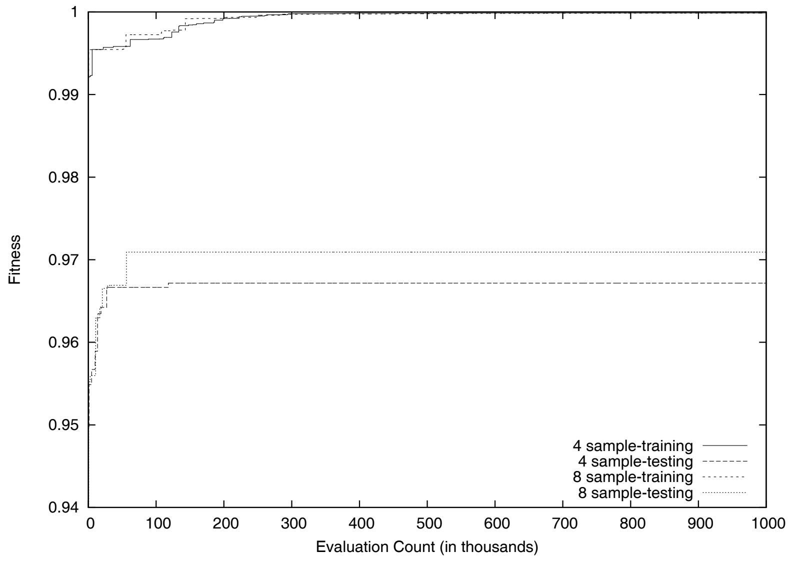

Figure 3: Overfitting in 4 station BAP. A GP metamodelling run is performed only with 4 and 8 samples until algorithm is terminated. Figure depicts that after random content is settled, validation fitness stagnates, while training converges to 1.0.

3.2 Buffer Allocation Problem

The buffer allocation problem (BAP) addresses the efficient utilisation of storage spaces in manufacturing systems. It emerges in many industrial systems such as supply chains operations and communication technologies (Dolgui, Eremeev, and Sigaev 2007) when there is limited availability of storage space. Intermediate buffers can enhance the efficiency and smooth operation of systems by eliminating disruptive effects of possible stochastic elements such as processing times and failures which are very common in many real-world operations. The reader is referred to Conway et al. (1988) for further illustration of the roles of buffers. Kim and Lee (2001) study the effect of buffer allocation schemes on work-in process. The literature reveals analytical queueing network models (Huang et al. 2002), Markov chain analysis (Papadopoulos et al. 1989) and simulation (Can et al. 2008) as example evaluative tools. Analytical techniques can be advantageous since performance criteria is measured in fractions of a minute. However, it can be tedious to derive such models for complex systems. Conversely, simulation facilitates a faster way of constructing more realistic models. However, its drawback appears in the evaluation effort when large scale, stochastic problems are considered as mentioned in Section 1. In our study, we will address this trade-off via constructing analytical surrogate models based on DES models (see Section 2.2).

In order to investigate the performance of genetic programming in constructing approximate functions of throughput rate of serial production lines as a function of buffer allocations. Figure 4 exemplifies a serial production line (flow line) of single machine servers, M_i, and finite intermediate buffers, B_i, of size q_i. The machines are modelled to have an exponential service rate with $\mu_i = 1$ in order to reflect the stochastic nature of the manufacturing plant.

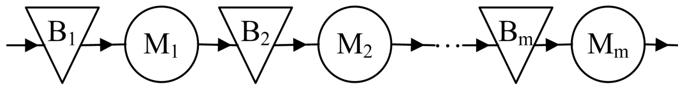

Figure 4: Serial production line with materials arriving to first buffer and products leaving from the last machine

In a production line as in the above figure, jobs arrive at the system at the first buffer and sequentially proceed through the line. As a result of having finite buffers and variability in processing times, starvation and blocking of the machines can be observed at intermediate stages. It is further assumed that the first machine is never starved and the last machine is never blocked similar to (Papadopoulos, Heavey, and Kelly 1989). Different ways of these dynamics are illustrated in Papadopoulos et al. (1993).

This study particularly considers BAP owing to the difficult nature of the problem. First, BAP occurs in a combinatorial discrete decision space with a large number of allocation scenarios (6). Most experimental design procedures are used for continuous domains. The need for discretisation of the generated samples can undermine design criteria considered by the procedure and prohibit its efficiency. The following subsection presents the comparison of DOEs on different sizes of BAPs.

$$C\begin{pmatrix} N+m-2 \\ m-2 \end{pmatrix} \tag{6}$$

3.3 Results

In this part, we will present the results of the metamodelling process in different sizes of BAP. We have tested our performance against single stage procedures, Latin Hypercube Sampling (LHS) and uniform design (UD) both with 100 samples for training and required discretisation of decision parameters. Each experiment of the framework is initialised with 4 experimental designs. In each experiment, we utilised the same validation data of randomly generated 80 samples which are not in the training set. The GP algorithm utilises the functional primitives, $\{$*,%,+,-, exponent, sqrt, cos, sin,x^y, ln, tan, $N(0,\sigma^2)\}$. The parameteres regarding GP and PSO are given in Tables 1 and 2.

Table 1: Parameter settings for GP

GP Parameters	Value
Pop Size	1000
Min Initial Tree Size	2
Max Initial Tree Size	4
Max Solution Length	20
Max Generation	100
Initialisation Method	Ramped Half and Half
Evaluation Limit	10^6

Table 2: Parameter settings for PSO

PSO Parameters	Value
Swarm size	20
Inertia weight	2.5
c_p, c_g	2.5
r_p, r_g	$\in$ [0,1]

Figure 5 depicts a sample run for problems in Table 3. Table 4 shows average size of training set in three experiments. PSO can provide an advantage in computational overhead particularly when model execution is time consuming.

Table 3: BAPs metamodelled

No. of Stations	Max. Buffer
4	32
9	27

Table 4: Average size of training sets

DOE	4 station	9 station
PSO	28	47
UD	100	100
LHS	100	100

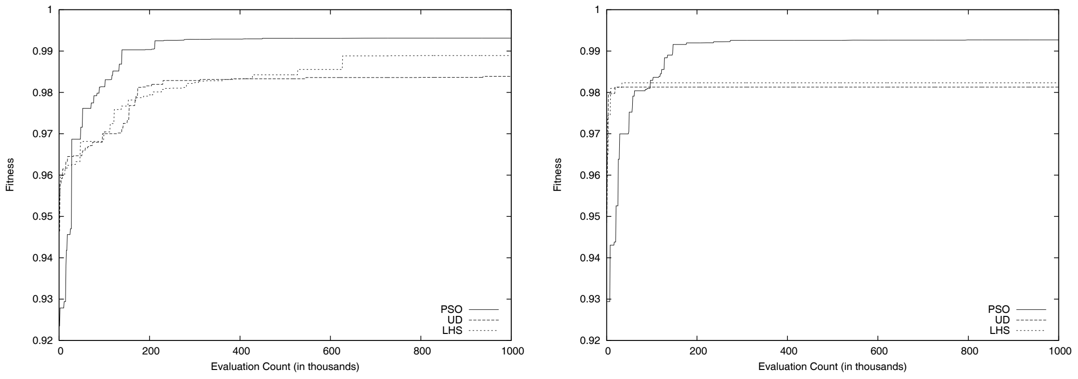

Figure 5: Metamodelling performance on 4 (left) and 9 (right) station problems

4 DISCUSSION AND FUTURE WORK

In this article, we have presented an evolutionary framework for simulation-based metamodelling. The framework utilises genetic programming and particle swarm optimisation to perform symbolic regression of simulation models based on sequential design of simulation experiments adaptively.

The empirical results present a gain in terms of computational overhead considering the effort in design of experiments. Particle swarm optimisation improved the efficiency of the procedure by resampling when overfitting starts. It was also useful in tackling the limitations considering the discrete variable space. Moreover, the genetic programming achieved an accuracy of at least 98% proximitiy to the actual simulation model regardless of the sampling technique.

In addition to the above points, PSO algorithm can also facilitate optimisation of BAP as it readily explores the decision space with the current state used in this article. It could be interesting to consider it to perform optimisation of the system. However, this discussion is beyond the scope of this work.

Finally, an important aspect we have not dealt in this article is the stochastic nature of the simulation output and multi-response metamodelling. We have planned to consider these in the near future.

ACKNOWLEDGMENTS

This research is funded by IRCSET, Irish Research Council for Science, Education and Technology.

REFERENCES

Alkhamis, T. M., and M. A. Ahmed. 2005. Simulation-based optimization for repairable systems using particle swarm algorithm. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 857–861. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Altiparmak, F., B. Dengiz, and A. A. Bulgak. 2007. Buffer allocation and performance modeling in asynchronous assembly system operations: An artificial neural network metamodeling approach. *Applied Soft Computing* 7 (3): 946–956.

Balasko, B., J. Madar, and J. Abonyi. 2006. Additive sequential evolutionary design of experiments. In *Artificial Intelligence and Soft Computing, ICAISC 2006*, 324–333. Springer.

Banks, A., J. Vincent, and C. Anyakoha. 2007. A review of particle swarm optimization. part i: background and development. *Natural Computing* 6 (4): 467–484.

Beers, W. C. M. V., and J. P. C. Kleijnen. 2004. Kriging interpolation in simulation: A survey. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 113–121. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Can, B., A. Beham, and C. Heavey. 2008. A comparative study of genetic algorithm components in simulation-based optimisation. In *Proceedings of the 2008 Winter Simulation Conference*, ed. S. J. Mason, R. R. Hill, L. Moench, O. Rose, T. Jefferson, and J. W. Fowler, 1829–1837. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Can, B., and C. Heavey. 2009. Metamodelling of discrete-event simulation models with genetic programming. *Submitted to Applied Soft Computing*.

Conway, R., W. Maxwell, J. O. McClain, and L. J. Thomas. 1988. The role of work-in progress inventory in serial production lines. *Operations Research* 36 (2): 229.

Dolgui, A., A. Eremeev, and V. Sigaev. 2007. Hbba: hybrid algorithm for buffer allocation in tandem production lines. *Journal of Intelligent Manufacturing* 18 (3): 411–420.

Foreman, N., and M. Evett. 2005. Preventing overfitting in GP with canary functions. In *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ed. H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, Volume 2, 1779–1780. Washington DC, USA: ACM Press.

Hendrickx, W., D. Gorissen, and T. Dhaene. 2006. Grid enabled sequential design and adaptive metamodeling. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, B. Lawson, J. Liu, and F. P. Wieland, 872–881. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Huang, M. G., P. L. Chang, and Y. C. Chou. 2002. Buffer allocation in flow-shop-type production systems with general arrival and service patterns. *Computers & Operations Research* 29 (2): 103–121.

Jin, R., W. Chen, and T. W. Simpson. 2001. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization* 23 (1): 1–13.

Kashan, A. H., and B. Karimi. 2009. A discrete particle swarm optimization algorithm for scheduling parallel machines. *Computers & Industrial Engineering* 56 (1): 216 – 223.

Kennedy, J., and R. Eberhart. 1995. Particle swarm optimization. In *1995 IEEE International Conference On Neural Networks*, Volume 1-6, 1942–1948. IEEE, Neural Networks Council: IEEE.

Kim, S., and H. J. Lee. 2001. Allocation of buffer capacity to minimize average work-in-process. *Production Planning and Control* 12:706–716.

Kleijnen, J., and W. v. Beers. 2003. Application-driven sequential designs for simulation experiments: kriging metamodeling. Discussion Paper 33, Tilburg University, Center for Economic Research.

Kleijnen, J. P. C., S. M. Sanchez, T. W. Lucas, and T. M. Cioppa. 2005. A user's guide to the brave new world of designing simulation experiments. *Informs Journal on Computing* 17 (3): 263–289.

Koza, J. 1992. *Genetic programming: On the programming of computers by means of natural selection*. Massachusetts: The MIT Press.

Li, G., S. Azarm, A. Farhang-Mehr, and A. R. Diaz. 2006. Approximation of multiresponse deterministic engineering simulations: a dependent metamodeling approach. *Structural and Multidisciplinary Optimization* 31 (4): 260–269.

Liu, B., L. Wang, and Y.-H. Jin. 2008. An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research* 35 (9): 2791–2806.

Murphy, G., and C. Ryan. 2008. A simple powerful constraint for genetic programming. In *Genetic Programming*, 146–157. Springer.

Papadopoulos, H. T., C. Heavey, and J. Browne. 1993. *Queuing theory*. Springer.

Papadopoulos, H. T., C. Heavey, and M. E. J. O. Kelly. 1989. Throughput rate of multistation reliable production lines with inter station buffers. (i) exponential case. *Computers in Industry* 13 (3): 229–244.

Poli, R., and N. McPhee. 2008. Parsimony pressure made easy. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ed. M. Keijzer, G. Antoniol, C. B. Congdon, K. Deb, B. Doerr, N. Hansen, J. H. Holmes, G. S. Hornby, D. Howard, J. Kennedy, S. Kumar, F. G. Lobo, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, J. Pollack, K. Sastry, K. Stanley, A. Stoica, E.-G. Talbi, and I. Wegener, 1267–1274. Atlanta, GA, USA.

Prechelt, L. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks* 11 (4): 761–767.

Wang, G. G., and S. Shan. 2007. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design* 129 (4): 370–380.

AUTHOR BIOGRAPHIES

BIRKAN CAN is a full time research postgraduate in Enterprise Research Centre at the Department of Manufacturing & Operations Engineering at University of Limerick (UL). He received a BSc. in Metallurgical and Materials Engineering from Middle East Technical University (METU), Turkey, in 2005. After a conditional year in Production Management in Industrial Engineering (METU), he has transferred to the UL for his Ph.D. in simulation-based optimisation. His current research interests involve simulation modelling; multi-objective optimisation and metamodelling; particularly of manufacturing and supply chain systems using evolutionary algorithms. His email address is <Birkan.Can@ul.ie>.

CATHAL HEAVEY is a Senior Lecturer of Operations Management in the Department of Manufacturing & Operations Engineering at the University of Limerick. He is an Industrial Engineering graduate of the National University of Ireland (University College Galway) and holds a M. Eng.Sc. and Ph.D. from the same University. He has published in the areas of queuing and simulation modelling. His research interests includes, simulation modelling of discrete-event systems; modelling and analysis of supply chains and manufacturing systems; process modelling; component-based simulation and decision support systems. His email address is <Cathal.Heavey@ul.ie>.