EVOLUTIONARY METAMODELLING OF DISCRETE–EVENT SIMULATION MODELS

Birkan Can[(a)], Gearoid Murphy[(b)], Cathal Heavey[(c)]

[(a)] Enterprise Research Centre, University of Limerick
[(b)] Biocomputing and Developmental Systems, University of Limerick

[(a)] Birkan.Can@ul.ie, [(b)]Gearoid.Murphy@ul.ie, [(c)] Cathal.Heavey@ul.ie

ABSTRACT
In this work, we investigate evolutionary metamodelling of discrete-event simulation models with the buffer allocation problems. We propose a genetic programming approach in order to derive the artificial response functions of simulation models. Alternative to similar studies, we do not assume a form for the response function and perform symbolic regression analysis over simulation models of different sizes of serial production lines. We present a comparative analysis with another artificial technique, neural networks, to identify the efficiency and the performance of symbolic regression in deriving metamodels via simulation.

Keywords: simulation optimisation, metamodelling, genetic programming, symbolic regression, decision support, buffer allocation problem, neural networks.

1. INTRODUCTION

A system can be described as a mechanism embodying the relationships among the interdependent entities in order to perform certain objectives, accompanied by performance metrics. The study of interacting components and analysis of their contribution on the system performance requires descriptive tools. One such a tool, discrete-event simulation (DES), facilitates a flexible environment to model and investigate many systems dealt by Operational Research/Management Science. This way, it enables understanding of the system behaviour and its performance dependent on model configuration, which can be particularly useful in system design and optimisation.

A possible means of exploring the effects of the system designs on the performance is to apply trial-error approaches by manipulating the system parameters. However, many real-world problems are NP-complete, with a large number of alternative configurations (William, Jerzy and Bahill 2001). Therefore, 'what-if' type techniques may remain inefficient in identification of this effect (Simpson, Peplinski, Koch et al. 2001). Alternatively, metamodelling can be utilised to perform this task.

Furthermore, efficiency in obtaining the system performance is an important factor in analysis and improvement of systems. As introduced earlier, DES provides its user the ability of modelling a system in as much detail as desired. While this is an important aspect, model execution times can be long, e.g. for DES models of large and stochastic systems. Real world systems may be of large scale resulting in long simulation runs. Reflecting the stochastic nature of most DES models multiple simulation replications will be required. The efficiency in providing system performance appears to be particularly important in computation intensive tasks, e.g. design, optimisation (Li, Azarm, Farhang-Mehr et al. 2006). The required amount of simulation runs in such applications may further render exploration for better system options not only challenging, but also impractical. Alternatively, with some sacrifice from accuracy, metamodelling can be an appealing alternative in modelling the system behaviour to replace DES.

In this work, we will introduce an evolutionary approach, Genetic Programming (Koza 1992), for metamodelling of DES models. Genetic Programming (GP) is an evolutionary algorithm (EA) which has the ability to generate analytical models of the underlying training data via symbolic regression (SR). This opportunity is investigated first on different buffer allocation problems in serial production lines. Following, comparative analysis with Artificial Neural Networks (ANN) and complexity assessment of the results are presented to identify the performance and usability of SR in metamodelling of DES models.

The remainder of this paper is organized as follows. In the following section, the literature review accompanied with the brief description of the BAP is provided. Subsequently, related experimental work illustrating the efficiency of GP in response surface metamodelling of DES will be given in Section 3. Finally, discussion and future work conclude the article.

2. LITERATURE REVIEW

2.1. Buffer Allocation Problem

This study considers serial production lines (flow lines) of single machine servers, $\mathbf{M_i}$, and finite intermediate buffers, $\mathbf{B_i}$, of size $\mathbf{q_i}$, as shown in Figure 1, for m-station serial line. The machines are modelled to have

exponential service rate with $\mu_i=1$ in order to reflect the stochastic nature of the manufacturing plant.

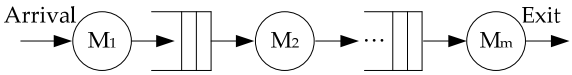

Figure 1: Serial production line.

The model of interest can be explained as follows. 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 can be observed at intermediate stages. When a machine is free to process, it takes a job from its upstream buffer. If there is no job to process in its upstream buffer, then the machine is said to be starved. Conversely, blocking in production lines can be commonly observed in two different ways, communication and production blocking (Papadopoulos, Heavey and Browne 1993). In our model, we assume that only production blocking occurs. In this type of blocking, a free machine takes a job from its upstream buffer for processing and passes it on to the downstream buffer after completion. However, if there is no space available in the buffer, the machine gets blocked and can not process new jobs. It is further assumed that the first machine is never starved and the last machine is never blocked.

Considering the illustrated processing scheme above, buffers can help improve efficiency and smooth operation of a manufacturing facility by eliminating disruptive effects of possible stochastic elements such as processing times and failures. The roles of buffers in manufacturing plants are further illustrated by Conway and Maxwell et al. (1988). The buffer allocation problem (BAP) addresses the efficient utilisation of storage spaces in manufacturing systems. It entails identifying the optimal allocation schemes to increase the performance of the operations. Besides manufacturing plants, similar problems can be observed in most supply chains operations and communication technologies (Dolgui, Eremeev and Sigaev 2007). Therefore, BAP has received a significant attention in research and practice, particularly in production and operations management.

The literature identifies the different aspects of production lines studied via buffer allocation. Kim and Lee (2001) investigate minimisation of work-in-process inventory (WIP) via buffer allocation, satisfying a minimum production rate (throughput rate) and an allowable buffer space. Similarly, Nahas and Ait-Kadi et al. (2006) and Diamantidis and Papadopoulos (2004) consider throughput rate as the performance criteria. Alternatively, Andijani and Anwarul (1997) evaluate the buffer allocation schemes via combining of WIP, cycle time and production rate. To obtain the performance data of a design, different tools such as; analytical queuing network models (De Almeida and Kellert 2000), Markov chain analysis (Vidalis, Papadopoulos and Heavey 2005) and simulation (Bulgak 2006) can be utilised. Analytical models are advantageous in the sense that system evaluations require less computation time. However, Altiparmak and Dengiz et al. (2007) indicate that they may suffer from being approximate models. Moreover, unrealistic assumptions may also be necessary to make the problem more tractable. An advantage of simulation over its analytical counterparts is its ability to reflect the system realities as much in detail as desired. However, its drawback appears in the time-cost of the evaluation of real-world systems as pointed out in Section 1. This may negatively effect the computational intensive applications involving the exploration of different alternatives, e.g. design, optimisation. To illustrate, despite the numerous optimisation techniques developed (Fu 2002, Tekin and Sabuncuoglu 2004), long simulation runs to evaluate the possible system alternatives may still render the overall process time-consuming. In such cases, approximate approaches can be alternatively used to estimate the response of a simulated system and to observe the effects of changes in model parameters. In the following section, a review of metamodelling in simulation is given.

2.2. Metamodelling in Simulation

It has been previously mentioned that metamodelling techniques can be appealing alternatives in modelling the system behaviour to replace DES with some sacrifice from accuracy when appropriate. There have been studies (Li, Azarm, Farhang-Mehr et al. 2006, Yang, Ankenman and Nelson 2007), in which simulation is coupled with metamodelling for more efficient performance measurements in the expense of the accuracy. Similarly, in this study a symbolic regression approach is taken in order to investigate metamodelling of the expected throughput rate dependent on the buffer allocation scheme.

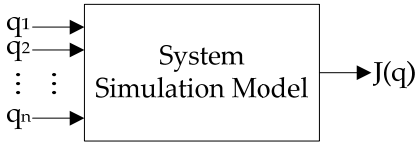

Figure 2: Simulation input-output relationship

A system, or particularly a simulation model, can be summarised as shown in Figure 2. Parameters; $\mathbf{q}$ is an input vector whose indices correspond to individual $\mathbf{q_i}$'s defining the decision variables of the system, where as $\mathbf{J_i(q)}$'s represent the performance criterion for a given configuration. In this work, $\mathbf{q}$ represents a buffer allocation scheme and expected throughput rate is the only performance measure considered.

Metamodelling can be defined as attaining an approximate function for $\mathbf{J_i(q)}$'s dependent on $\mathbf{q_i}$'s in order to understand behavioural aspects of the systems, e.g. estimation of the change in performance induced by the change in the configuration. It involves constructing approximate analytical models representing this function as in Equation 1, in which $\mathbf{q}$ is the system configuration and $\boldsymbol{\varepsilon}$ is the error in approximation.

$$\hat{J}(q) = f(q) \text{ such that } J(q) = \hat{J}(q) + \varepsilon \qquad (1)$$

The main concern in metamodelling is efficiently obtaining a reasonably accurate global model of the system (or a simulation model) (Wang and Shan 2007). While efficiency requires the minimum computational effort in attaining the metamodel, accuracy will reflect how close the metamodel is to the system. Generally, a metamodelling study is consisted of three major steps:

1. Sampling the data,
2. Modelling the collected data,
3. Fitting the model to sampled data.

Sampling refers to collecting sufficient data to reflect the system behaviour, i.e. attaining system performance at prespecified design points. Model choice and fitting are the subsequent steps in a metamodelling study. Briefly, the response is assumed to take a form, while this form can be a one or higher order polynomial dependent on the decision variables, network of neurons, decision trees or nonlinear transformations can be used as alternatives. There have been many techniques developed to perform these major steps above. Batmaz and Tunali (2002) give a comparative study considering sampling. Wang and Shan (2007) gives an extensive overview of the field outlining the roles and benefits of metamodelling. Simpson and Peplinski et. al (2001) present some of the popular methods, among which Polynomial Regression (PR), Kriging (KG), Artificial Neural Networks (ANN) have been prominently utilised methods in metamodelling DES. Jin, Chen and Simpson (2001) present a systematic comparison of common techniques on problems of different scales and difficulties.

There is a variety of metamodelling applications considering DES. In a recent study, Biles and Kleijnen et. al (2007) present a simulation-based optimisation study of constrained (s,S) inventory systems using KG. Noguera and Watson (2006) apply PR to attain the metamodel of a chemical plant as a function of capacity and throughput. Similarly, Durieux and Pierreval (2004) use second order polynomials to perform sensitivity analysis of a flexible manufacturing system to outline the effects of design parameters on average system utilisation. Afonin and Derjabkina et. al (2007) exemplify the use of ANNs in analysis of complex systems via metamodelling.

There have also been attempts exploiting ANNs to assist system evaluation in problems considering BAP. Artificial Neural Networks (ANNs), inspired by nature, emulate the nervous systems, such as human brain. As the interaction of neurons provides information processing capabilities, such as learning, ANNs can be trained over a set of data sample to generate the metamodels of simulated systems. Chambers and Mount-Campbell (2002) employ ANNs in a queuing system to derive product throughput rates and average system sojourn time to identify the optimum buffer sizes. Altiparmak, Dengiz and Bulgak (2007) investigate predictive capabilities of ANNs in assembly systems against polynomial and exponential regression models. Similarly, Bulgak (2006) uses ANNs in simulation metamodelling to obtain production rate of an assembly system in optimisation via buffer allocation. Finally, in a recent study, Person, Grimm and Ng (2008) exploits ANNs to assist optimisation routine in order to avoid long simulation runs.

The above discussions indicate that metamodelling can be employed as an efficient tool to obtain reasonably accurate global approximations to DES models. Such global models can contribute to understanding of the system behaviour based on the decision parameters. The literature review implies that there is still room for further research. In the following section, genetic programming will be introduced in order to perform based metamodelling of DES models based on symbolic regression.

2.3. Symbolic Regression with Genetic Programming

Evolutionary algorithms (EAs) are meta-heuristic techniques that are developed to solve difficult non-linear problems. In analogy to evolution in nature, EAs implement its operators and processes. The process of evolution imitated by EAs can be summarised as in the below pseudo-code:

1. Generate initial population, P(t);
2. Evaluate the individuals in P(t);
3. Repeat
    a. Select parents from P(t) to reproduce;
    b. Generate new offspring via crossover and mutation
    c. Evaluate the new individuals and insert to next generation;
4. Until termination.

The abstract definition of EAs as shown above has led to many sub-branches. Genetic Programming (GP) is an EA which can be used to evolve programs (Koza 1992). These programs may be interpreted as mathematical expressions, building instructions, grammar rules, etc. GP can generate symbolic representations of the training data which are often simply mathematical expressions, functions of the application domain.

Traditionally, regression refers to the use of numeric techniques to solve for the error against given training data, such as the back propagation algorithm of ANNs. Such approaches require prudent selection of configuration parameters to protect against the phenomenon of overfitting (Tetko, Alexander and Luik 1995), where the complexity of the model inhibits its performance on unseen data. Furthermore, the resulting set of numeric parameters are very difficult to interpret meaningfully for practitioners.

The use of GP for generating symbolic representations of training data was first proposed by (Koza 1992). Such representations are often simply mathematical expressions, functions of the application

domain. Through the use of effective convergence algorithms, wherein the dynamics of evolution are enhanced via constraints, GP can produce compact expressions which have excellent generalisation properties (Murphy and Ryan 2008). These derived solutions have great potential for yielding insight into the underlying functional relationships of the application domain and have been used to aid geneticists in the comprehension of complex regulatory networks used in expressing DNA (Moore, Barney and White 2008).

A standard regression study involves determination of coefficients of a prespecified functional structure (generally a low order polynomial) which explains dependency of a parameter to independent design variables (Simpson, Peplinski, Koch et al. 2001). In contrast, symbolic regression does not require a strict specification of the size and shape, in other words, structural complexity of the solution. This stems from the fact that it requires a function set to evolve instead of assuming a compact analytical form. This implies that the functional form and its coefficients are explored simultaneously in symbolic regression to obtain response functions (Koza 1992).

This study uses these properties of GP to generate accurate symbolic models of DES which are potentially of great use in both optimising the processes and in enhancing the confidence practitioners have in using such artificially derived solutions by virtue of the succinct expressions produced by GP.

GP needs several important components to perform symbolic regression. GP uses a tree-based representation to search the space of solutions (see Figure 3.

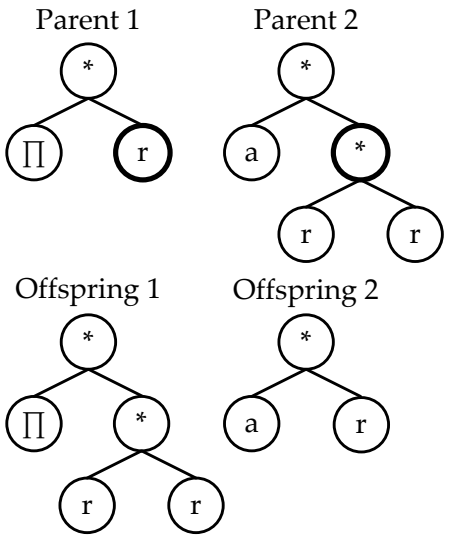

Figure 3: Tree-based representation and an example of crossover operation in GP. Crossover points on the parents are indicated by bold circles.

The traditional concepts of variation operators, i.e. recombination (crossover) and mutation in EAs, are adapted to the particular context of the GP representation, exemplified in the form of subtree swap in Figure 3. Similarly, mutation operates asexually and can perform removing (or replacing) subtrees. A discussion on the role of variation operators can be found in Luke and Spector (1997). Further details regarding the GP implementation can be found in Section 3.

There have been a variety of applications of symbolic regression via genetic programming reported in the literature. Duffy and Warnick (1999) highlight the ability of GP in deriving functions implying strategies in decision-making. Castillo and Kordon et. al (2005) apply symbolic regression for statistical model building in industrial chemical processes. In another application, Korns (2007) investigates symbolic regression on large-scale complex problems and tries improving the efficiency of SR via integration of formal grammar rules to modelling process.

To conclude, the above discussions justify the use of GP in simulation-based metamodelling. The following section will present the experiments to assess the quality and usability of symbolic regression in metamodelling of DES.

3. EXPERIMENTS AND RESULTS

This section presents the results of a set of experiments in which the ability of GP to generate accurate metamodels of simulations is compared and contrasted against an ANN performing the same task. Both systems used the buffer size of the stations in the simulation as their input and attempted to predict the throughput rate of the simulation. The performance graphs show the testing performance of the systems, a good measure of generalisation ability. The performance of the techniques are represented via root mean squared error between the predicted throughput and the actual throughput as 1/(1+RMSE). All results shown here are the results of 100 independent runs.

The testing and training data was derived by generating a random distribution of buffer sizes amongst the stations of the model and evaluating the resulting throughput rate for that buffer configuration. Two sets of 100 buffer samples were used for testing and training on each simulation model. We used simulation models consisting of 4, 8, 12, 16 and 20 stations with a maximum buffer size of 20.

The GP algorithm used here employs a variation of the Hereditary Repulsion (HR) convergence manipulation protocol to enable the GP evolve compact and powerful expressions (Murphy and Ryan 2008). The algorithm uses a generational framework in which the next generation is produced entirely from the current generation. Crossover events cannot span more than one generation. The initial population is created using the ramped half and half initialisation method (Koza 1992). This produces a broad spread of expressions of varying size. Each expression is given a fitness derived from the RMSE of its ability to predict throughput; 1/(1+RMSE).

Random selection is used to pick parents from the population. This selection strategy increases performance by reducing the pressure on the population to converge, thus allowing evolution more time to

discover useful functionality. Once an offspring from the random parents has been evaluated, it is only allowed into the next generation if it is better than *both* parents. Should the offspring fail this criterion, it is discarded and the process begins again. It is not possible for a parent to move from the current generation to the next generation. It must preserve its genes by combining with another solution in a manner superior to both parents.

Because of the potential for failed crossovers, the number of evaluations per generation is variable with the HR algorithm; therefore it must be compared to other systems in terms of evaluations. The functional primitives used in the GP algorithm are as follows; (*, %, +, -, exponent, sqrt, cos, sin). Subtree crossover was used without any mutation, as the HR algorithm does not require it. The parameters used to initialise the GP settings are described in Table 1.

Table 1: GP Parameters

Parameters	Value
Population Size	1000
Min Initial Tree Size	2
Max Initial Tree Size	4
Max Tree Size	12
Initialisation Method	Ramped Half and Half
Evaluation Limit	10^6

The Multi-Layer Perceptron (MLP) used a standard back-propagation algorithm to perform learning (Rumelhart, Hinton and Williams 1986). Its parameters are described Table 2.

Table 2: MLP Parameters. These parameters were found to consistently give the best MLP performance across all the experiments.

Parameters	Value
Hidden Layer Size	10
Learning rate	0.01
Initialisation Range	+/-1

Table 3: GP vs. ANN testing fitness results giving a statistical confidence of %99.9 on GP performance surpasses the ANN for the given configurations.

Simulation	GP Fit	T-Value	MLP Fit
4 Station	0.944215	-2.236917	0.945442
8 Station	0.939218	30.427879	0.928591
12 Station	0.941697	52.115238	0.933142
16 Station	0.943745	22.775646	0.939181
20 Station	0.947534	29.964231	0.941968

The results are tabulated in Table 3. The Students T-Test was evaluated for each experiment. Table 3 clearly shows that GP outperforms the MLP for all experiments except the easiest 4 station configuration. In absolute terms however; the performance difference between the paradigms is slight. Despite the slim difference between GP and the MLP, it was observed consistently in the experiments and produced high statistical confidence values.

A figure illustrating the regression performance for both GP and the MLP over a duration of 10^6 evaluations is given in Figure 4. This shows that the MLP quickly converges but fails to progress any further. By contrast, GP is slow to converge but manages to successfully sustain convergence, finding higher quality solutions than the MLP.

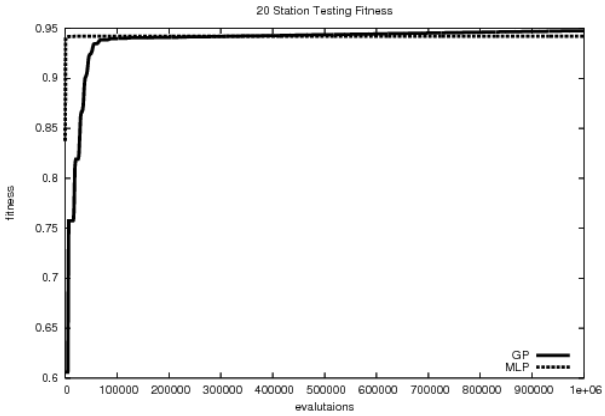

Figure 4: Graph illustrating contrasting the regression dynamics of MLP and GP for the 20 station problem. MLP quickly converges but GP eventually surpasses the performance of the MLP.

A clear sense of the difference between the GP and MLP emerges when one examines the complexity of the resulting solutions. We measure its complexity as the number of components involved in contributing to the solution. For the MLP, this is the number of weights in the network. For the 4 station experiment the MLP consisted of 50 weights, 4 input nodes by 10 hidden nodes plus the final 10 weights for the output nodes. Similarly, for the 8, 12, 16 and 20 station experiments, the MLP consisted of 90, 130, 170 and 210 components.

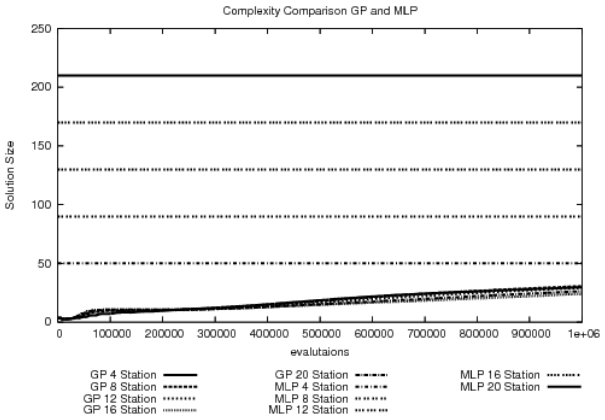

Figure 5: Fixed size MLP solutions are far more complex than the GP solutions.

By contrast, GP consistently utilises a small fraction of the components used by the MLP, illustrated in Figure 5. These smaller compact solutions

immediately provide a viable platform for analysing the dynamics of the simulation model. Table 4 shows in detail the actual average sizes of evolved solutions in the GP population.

Table 4: Table illustrating the contrast in size of solutions between GP and MLP.

Simulation	GP Tree Node Size	MLP Weights Size
4 Station	29.6925	50
8 Station	30.5628	90
12 Station	29.0962	130
16 Station	24.0059	170
20 Station	26.1847	210

4. DISCUSSION AND CONCLUSIONS

Discrete-event simulation provides a flexible modelling platform enabling the detailed analysis of many industrial systems. However, long model execution time particularly is a common challenge in simulation-based computation intensive studies, such as design and optimisation. In this work, we have tried to address this issue with symbolic regression via genetic programming (GP) by generating analytical approximations with %0.06 proximity to the actual model behaviour.

In order to assess the applicability of genetic programming in metamodelling of simulation, serial production lines with buffer allocation problem were studied to identify dependency of average expected throughput rate on the individual buffer allocations. We have compared our results against ANNs, which are prominently applied in metamodeling of buffer allocation problem. The two techniques were analysed in terms of the quality of approximation and complexity of the generated models. It has been statistically shown that GP performance is not worse than the implemented ANN for the given test problems in terms of the accuracy of the approximation. Moreover, GP has demonstrated a tendency towards surpassing the performance of the ANN. We believe, this ability can be further improved with the use of appropriate sampling techniques to attain the training data, whereas randomly selected design points are utilised in our experiments.

Furthermore, a clear sense of the difference between GP and ANN emerges when the complexity of the resulting solutions are examined. Our results have illustrated that GP consistently utilises a small fraction of the components used by the ANN. These smaller compact solutions immediately provide a viable platform for analysing the dynamics of the simulation model. This fact implies a considerable use to the practitioner as GP has provided a tangible functional decomposition of the simulation. In this respect, symbolic regression via GP can allow a window into the `black box' of the simulation.

GP has been observed to exhibit latency at the initial stage of the experiments which is a consequence of the system expending a great effort to improve the initial random content of the population. However, this latency is a negligible drawback considering the fact that time cost of obtaining a metamodel is in fractions of attaining training data. Nonetheless, we would like to focus on this value loss in our future studies.

Finally, an interesting observation deserves to be pointed out in model complexity. The complexity analyses indicate that GP consistently produces smaller solutions despite the problem size getting larger. This may be due to the fact that effect of certain buffer positions in the serial line may be more significant than others in describing the throughput rate of the model. While this leads GP to generate less complex expressions, it actually locates the bottleneck positions through the production line as well. This observation will need further analysis in future work.

ACKNOWLEDGMENTS

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

REFERENCES

Afonin, P., V. Derjabkina, A. Kozhukhova and O. Lamskova (2007). "The design and analysis of complex system neural network metamodels." *Optical Memory \& Neural Networks* **16**(3): 154-158.

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.

Andijani, A. A. and M. Anwarul (1997). "Manufacturing blocking discipline: A multi-criterion approach for buffer allocations." *International Journal of Production Economics* **51**(3): 155-163.

Batmaz, I. and S. Tunali (2002). "Second-order experimental designs for simulation metamodeling." *Simulation-Transactions of the Society for Modeling and Simulation International* **78**(12): 699-715.

Biles, W. E., J. P. C. Kleijnen, W. C. M. van Beers and I. A. van Nieuwenhuyse (2007). Kriging metamodeling in constrained simulation optimization: an explorative study

Kriging metamodeling in constrained simulation optimization: an explorative study. *Simulation Conference, 2007 Winter*.

Bulgak, A. A. (2006). "Analysis and design of split and merge unpaced assembly systems by metamodelling and stochastic search." *International Journal of Production Research* **44**: 4067-4080.

Castillo, F., A. Kordon, J. Sweeney and W. Zirk (2005). Using Genetic Programming in Industrial Statistical Model Building. Genetic

Programming Theory and Practice II, Springer US**:** 31-48.

Chambers, M. and C. A. Mount-Campbell (2002). "Process optimization via neural network metamodeling." *International Journal of Production Economics* **79**(2): 93-100.

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.

De Almeida, D. and P. Kellert (2000). "An Analytical Queueing Network Model for Flexible Manufacturing Systems with a Discrete Handling Device and Transfer Blockings." *International Journal of Flexible Manufacturing Systems* **12**(1): 25-57.

Diamantidis, A. C. and C. T. Papadopoulos (2004). "A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines." *Mathematical Problems in Engineering* **2004**(3): 209-223.

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.

Duffy, J. and J. Warnick (1999). Using Symbolic Regression to Infer Strategies from Experimental Data, Society for Computational Economics.

Durieux, S. and H. Pierreval (2004). "Regression metamodeling for the design of automated manufacturing system composed of parallel machines sharing a material handling resource." *International Journal of Production Economics* **89**(1): 21-30.

Fu, M. C. (2002). Optimization for discrete-event simulation: Theory and practice. System Simulation and Scientific Computing. Z. J. Chen. Beijing, Int Acad Publ Beijing World Publ Corp**:** 50-50.

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.

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

Korns, M. (2007). Large-Scale, Time-Constrained Symbolic Regression. Genetic Programming Theory and Practice IV**:** 299-314.

Koza, J. R. (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.

Luke, S. and L. Spector (1997). A Comparison of Crossover and Mutation in Genetic Programming. *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, USA.

Moore, J. H., N. Barney and B. C. White (2008). Solving Complex Problems in Human Genetics Using Genetic Programming: The Importance of Theorist-Practitionercomputer Interaction. Genetic Programming Theory and Practice V, Springer Berlin / Heidelberg**:** 69-85.

Murphy, G. and C. Ryan (2008). A Simple Powerful Constraint for Genetic Programming. *Genetic Programming*, Springer Berlin / Heidelberg**:** 146-157.

Nahas, N., D. Ait-Kadi and M. Nourelfath (2006). "A new approach for buffer allocation in unreliable production lines." *International Journal of Production Economics* **103**(2): 873-881.

Noguera, J. H. and E. F. Watson (2006). "Response surface analysis of a multi-product batch processing facility using a simulation metamodel." *International Journal of Production Economics* **102**(2): 333-343.

Papadopoulos, H. T., C. Heavey and J. Browne (1993). *Queuing Theory*, Springer.

Person, A., H. Grimm and A. Ng (2008). A Metamodel-Assisted Steady-State Evolution Strategy for Simulation-Based Optimization. *Trends in Intelligent Systems and Computer Engineering**:*** 1-13.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). "Learning representations by back-propagating errors." *Nature* **323**(6088): 533-536.

Simpson, T. W., J. D. Peplinski, P. N. Koch and J. K. Allen (2001). "Metamodels for computer-based engineering design: survey and recommendations." *Engineering with Computers* **17**(2): 129-150.

Tekin, E. and I. Sabuncuoglu (2004). "Simulation optimization: A comprehensive review on theory and applications." *IIE Transactions* **36**(11): 1067-1081.

Tetko, I. V., D. J. L. Alexander and I. Luik (1995). "Neural Network Studies. 1. Comparison of Overfitting and Overtraining." *Chemical Information and Modeling* **35**(5): 826-833.

Vidalis, M. I., C. T. Papadopoulos and C. Heavey (2005). "On the workload and 'phaseload' allocation problems of short reliable production lines with finite buffers." *Comput. Ind. Eng.* **48**(4): 825-837.

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.

William, L. C., R. Jerzy and A. T. Bahill (2001). "System design is an NP-complete problem: Correspondence." *Syst. Eng.* **4**(3): 222-229.

Yang, F., B. Ankenman and B. L. Nelson (2007). "Efficient generation of cycle time-throughput curves through simulation and metamodeling." *Naval Research Logistics* **54**(1): 78-93.