Design of Experiments for training metamodels in simulation-based optimisation of manufacturing systems

Anna Syberfeldt[1*], Henrik Grimm[1], Amos Ng[1]

[1]Center for Intelligent Automation University of Skövde Skövde, 541 48, Sweden

ABSTRACT

In this study, the use of DoE for training metamodels in simulation-based optimisation of manufacturing systems is evaluated. The evaluation is done through a case study of a real manufacturing system. A simulation model of the system exist and the aim is to train an Artificial Neural Network as a metamodel of the system with as high accuracy as possible. Two training data sets generated using different DoE designs are evaluated and compared to a random training data set. The combination of DoE generated data and randomly sampled data is also evaluated.

1. INTRODUCTION

Real-world manufacturing systems often contain nonlinearities, combinatorial relationships and uncertainties. To face the increasing complexity of these systems, various modelling methods have been proposed during the last decades. Several of these methods use an analytical approach, for example queuing theory, Petri nets, and finite state automata. The main disadvantage of analytical methods is that modifications of the physical system imply lots of modelling work, since every time a change occurs, the model needs to be remade [1]. Many manufacturing systems are also too complex to be modelled in an analytical manner [2]. To overcome these problems, simulation has been proposed as a flexible method for systems modelling. Using a simulation model, the problem under consideration can be analysed and optimised in a convenient way (Fig. 1). A wide range of simulation applications for the manufacturing domain have been described in the literature, ranging from the overall system level down to low-level machine processes.

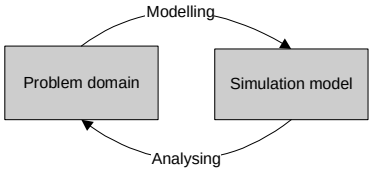

Figure 1: Systems modelling using simulation.

Although a simulation run provides the user with valuable knowledge about the system, a simple performance evaluation is often insufficient and a more exploratory process is often desirable for determine the optimal system settings. This need has motivated the technique of simulation-based optimisation (SO). In general terms, SO is the process of finding the best values of some parameters for a system, where the performance of the system is evaluated based on the output from a simulation model of the system [3]. In a manufacturing system, for example, one might be interested in finding the optimal buffer settings with respect to throughput of the system and average cycle time of products. Finding the optimal parameter values is an iterative simulation-optimisation process. In this process, an optimisation procedure first generates a set of input parameter values and feeds them to a simulation that estimates the performance of the system. Based on the evaluation feedback from the simulation, the optimisation procedure then generates a new set of input parameter values and the generation-evaluation process continues until a user-defined stopping criterion is satisfied.

* Corresponding author: Tel.: (46) 500-448577; E-mail: anna.syberfeldt@his.se

A variety of successful applications of SO have been reported in the manufacturing domain. However, in spite of the great success achieved in many applications, SO has also encountered some challenges. Typically, thousands of simulation evaluations are required and one single evaluation may take a couple of minutes to hours of computing time. This poses a serious hindrance to the practical application of SO in real-world scenarios. Time-consuming simulations may be tolerated in the design phase, but is not acceptable in the daily operational optimisation. To address the problem of time consuming simulation, computationally efficient metamodels have been suggested (Fig. 2). Metamodels approximate the relationship between the simulation input and output parameters by computationally efficient mathematical models. In the context of SO, a simulation model can be viewed as a function that turns input parameter values into output performance measures [4] and the purpose of the metamodel is to represent this function.

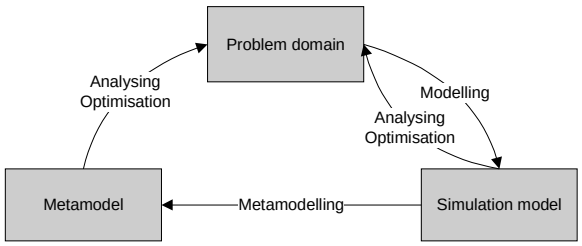

Figure 2: Using metamodelling in simulation-based optimisation.

A variety of different metamodelling techniques have been proposed, such as Kriging models, polynomal models, and Artificial Neural Networks (ANNs). In real-world applications which often have a high-dimensional design space and a limited number of data samples, ANN is the technique preferred [5]. By adopting ANNs in simulation, the computational burden of the optimisation process can be greatly reduced [6].

ANNs learn to approximate simulation input-output relationships through training. The user presents a number of simulated input-output samples to the network, which tries to represent the underlying function of the relationships [7]. In the literature, it has been suggested that the samples used to train the ANN should be derived from simulation runs based on the technique of Design of Experiments (see for example, [7]-[9]). In short terms, Design of Experiments (DoE) offers a way to perform structured simulation experiments through design plans that specifies simulation input settings [10]. The overall procedure for training ANNs to estimate simulations based on training data generated from DoE is simple [11]:

a) construct an experimental plan of the different simulation input parameter settings that are to be tested using DoE,

b) simulate the system with the given input parameter values, and

c) train the ANN using the simulated input-output samples.

In this study, we evaluate the technique of DoE for generation of training data in simulation metamodelling using ANNs. The evaluation is done through a case study of a real manufacturing system. A simulation model of the system exist and the aim is to train an ANN as a metamodel of the system with as high accuracy as possible. Two training data sets generated using different DoE designs are evaluated and compared to a random training data set. The combination of DoE generated data and randomly sampled data is also evaluated.

2. BACKGROUND

In this section, the concepts of DoE and ANNs are explained.

2.1 DESIGN OF EXPERIMENTS

In DoE, a design is a matrix where every column corresponds to a simulation input parameter and the entry in a column corresponds to a setting for this parameter. A simulation input parameter is called a *factor* and an output parameter is called a *response*. The purpose of a design is to indicate how the factors should be varied to study their impact on the response. A factor can be either *qualitative* or *quantitative*. For example, in a manufacturing factory a qualitative factor might be operator shift system (two shift, three shift, etc.), and a quantitative factor might be the

number of operators working during a shift. A factor can be set to two or more values, called *levels*. A row in the matrix represents a combination of levels and is called a *design point* (sometimes also called *scenario*). Levels are usually coded in a standard way to allow designs to be reused. For quantitative data, the most common code is to specify low and high levels as -1 and +1. An example of a design matrix is shown in Table 1.

Design point	Parameter 1	Parameter 2
1	-1	+1
2	-1	-1
3	+1	-1
4	+1	+1

Table 1: Example of a design.

A lot of different designs have been suggested in the literature. In this work, we focus on three designs that have been considered particularly useful in simulation [10]. These are *2^k factorial design*, *m^k factorial design*, and *Latin hypercube design*.

2.1.1 2^K FACTORIAL DESIGN

The 2^k factorial design specifies two levels for each of *k* factors, normally -1 and +1 (representing the low and high setting of a factor, respectively). Fig. 3 shows an example of a 2^2 factorial design illustrated as sampling the corners of a hypercube, defined by the low and high values of the two design parameters (P1 and P2). The same 2^2 factorial design is also shown in Table 1.

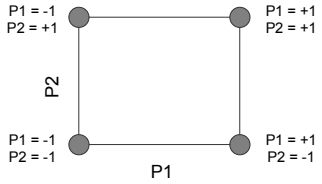

Figure 3: 2^2 Factorial Design.

The 2^k factorial design is simple to implement and is one of most commonly used factorial design [10]. It makes it possible to examine several factors at a time and to analyse interaction effects between design parameters. However, although the 2^k factorial design is useful it has some limitations. Since only two levels of each factor are examined, only the corners of the experimental region is examined and the user is not provided with any information about what happens in the centre of the region. To address this problem, the m^k factorial design has been proposed, defining sample points in a finer grid of the experimental region.

2.1.2 M^K FACTORIAL DESIGN

The m^k factorial design (also called full factorial design) defines *m* number of levels for each of *k* design parameters. A larger value of *m* means better space filling properties and more information about the system compared to a 2^k factorial design [10]. Fig. 4 (inspired by [10]) illustrates the difference between the designs, comparing a 2^2 and a 10^6 factorial design for an industrial assembly robot with two design parameters: speed and precision. In both hypercubes green points represent successful results, red points represent unsuccessful results, and yellow points represent results in between these.

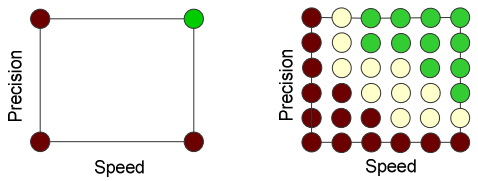

Figure 4: Comparison between 2^2 factorial design (left) and a 6^2 factorial design (right).

The only information provided by the 2^2 factorial design is that the robot is performs well when the values of both precision and speed are high. With the 6^2 factorial design considerable more details are provided. For example,

one might notice that a high precision of the robot is not necessary for successful results as long as maximum speed is maintained.

Although the m^k factorial design provides lots information about the system, it has also received some criticism. The design is only useful for a limited number of factors because of the exponential grow in data samples required. The need of more efficient experimental designs for many factors has given rise to a design technique called *Latin Hypercube Sampling*, which is described in the next section.

2.1.3 LATIN HYPERCUBE DESIGN

The Latin hypercube (LH) design requires considerable less data sampling compared to the full factorial design, but still have good space-covering properties. In a LH design, each of the factors is sampled exactly once at each level. Fig. 5 (inspired by [10]) shows an example of such random sampling for the industrial assembly robot discussed in the previous section.

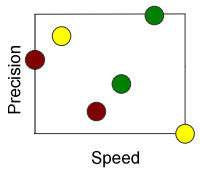

Figure 5: Latin hypercube design.

In comparison with the 2^2 factorial design for the same problem (Fig. 4, left), the LH design provides some information about factor combinations in the centre of the experimental region. The LH design do not provide as much information as the 10^6 factorial design (Fig. 4, right) and the user can not be sure about the boundaries between the sub-regions of unsuccessful, successful, and mixed results. However, the user is still able to identify important properties of the system, such as for example that with a low precision and high speed the robot achieves ok results.

Especially when the number of factors increase, the benefits of LH designs become obvious. In LH designs, the number of design points grows linearly with the number of factors, rather than exponentially. An experiment with 30 factors, for example, requires 30 sampling points with an LH design, compared to over 10^9 sampling points for a 2^{30} factorial design. Returning to the assembly robot example, suppose that a simulation run of the robot takes 10 seconds – then the experiment would take 5 minutes with an LH design and over 300 years with 2^k factorial design.

2.2 ARTIFICIAL NEURAL NETWORK

The use of metamodels to reduce the limitations of time consuming simulations was first proposed by Blanning in 1975 [12]. Traditionally, regression and response surface methods have been two of the most common metamodeling approaches [13]. In recent years, ANN have gained increased popularity, as this technique requires fewer assumptions and less precise information about the systems being modelled when compared with traditional techniques [5].

In general terms, an ANN is a non-linear statistical data modelling method used to model complex relationships between inputs and outputs [7]. Originally, the inspiration for the technique was from the area of neuroscience and the study of neurons as information processing elements in the central nervous system. ANNs have universal approximation characteristics and the ability to adapt to changes through training. Instead of only following a set of rules, ANNs are able to learn underlying relationships between inputs and outputs from a collection of training examples, and to generalize these relationships to previously unseen data. These attributes make ANNs very suitable to be used as the substitutes for computationally expensive simulation models.

The first work providing the foundations for developing ANN metamodels for simulation was done by Pierreval [14] and Pierreval and Huntsinger [15] in 1992. Both of these studies yielded results that indicated the potential of ANNs as metamodels for discrete-event and continuous simulation, particularly when saving computational cost is important. Since then, many applications of ANN-based metamodels in simulation systems have been reported (see, for example, [11][16]-[17]).

3. A REAL-WORLD APPLICATION OF DESIGN OF EXPERIMENTS FOR METAMODELLING

This section describes an application of DoE for metamodelling of a real-world manufacturing system. The manufacturing system considered is present and the simulation model built of it is described. The concepts of the ANN constructed as a metamodel of the simulation is also described.

3.1 MANUFACTURING SYSTEM

The case study carried out in this paper is done on a manufacturing system at Volvo Aero. The largest activity at Volvo Aero is development and production of advanced components for aircraft engines and gas turbines. Nowadays, more than 80 percent of all new commercial aircraft with more than 100 passengers are equipped with engine components from Volvo Aero. Volvo Aero is also producing engine components to space rockets. As a partner of the European space program, they develop rocket engine turbines and combustion chambers.

At the Volvo Aero factory in Trollhättan, a new manufacturing cell has recently been introduced for the processing of engine components. The highly automated cell comprises multiple operations and is able to process several component types at the same time. After a period of initial tests, full production is now to be started in the cell. Similar to other manufacturing companies, Volvo Aero continuously strives for competitiveness and cost-reduction, and it is therefore important that the new cell is operated as efficient as possible.

3.2 SIMULATION MODEL

To aid the production planning, a simulation model of the cell has been built using the SIMUL8 software package. The simulation model provides a convenient way to perform what-if analyses of different production scenarios without the need of experimenting with the real system. Besides what-if analyses, the simulation model can also be used for optimisation. In this paper, we describe how the simulation model has been used to optimise the scheduling of components to be processed in the cell. For the production to be as efficient as possible, it is interesting to find a schedule that is optimal with respect to maximal utilisation of the cell in combination with minimal shortage and tardiness of components. Mathematically, the optimisation objective is described by

$$\sum_{i \in C} \left(w_s i_{shortage} + w_t i_{tardiness} \right) - w_u utilisation \qquad (1)$$

where C is the set of all components and w is the user-defined weighted importance of each objective. The goal of an optimisation process is to minimise the objective function value.

For the scenario considered in the paper, five component types are included and a duration corresponding to two weeks of production is simulated. The input of the simulation consists of a vector of five real values, corresponding to lead time for each component type. The values are bounded by a lower and upper bound, specific for each component type. The output produced by the simulation consists of a vector of four values, corresponding to utilisation, component shortage and total tardiness of components of components.

3.3 ARTIFICIAL NEURAL NETWORK

An ANN is constructed as a metamodel of the SIMUL8 simulation model. The ANN is trained to estimate the objective function value described in Equation 1 as a function of planned lead times of components. The ANN has feed-forward architecture with one hidden layer comprising three nodes (Fig. 6).

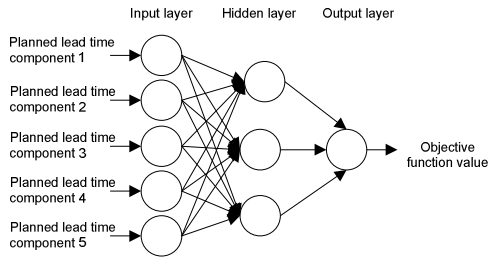

Figure 6: Illustration of Artificial Neural Network.

For training the ANN, two different data sets are generated using the technique of DoE. The first data set is generated using a full factorial design with three levels, i.e. a 3^5 factorial design. The second data set is generated using a LH design with 243 levels. A set of randomly sampled data is also generated as a baseline reference. All three data sets comprise the same number of data samples (243).

The ANN is trained with the three data sets separately using batch training and the RPROP algorithm [18]. Before training takes place, the data is linearly normalized to values between 0.1 and 0.9. To avoid overtraining, ten-fold crossvalidation is used during the training (for a description of *k*-folded crossvalidation, see [19]). The ANN is trained for a maximum of 1000 iterations for each fold.

4. RESULTS

The ANN trained with the three different data sets (full factorial, LH, and random) is evaluated in two aspects: how accurately it represents the simulation model, and how useful it is in an optimisation process.

4.1 ACCURACY

To evaluate the accuracy of the ANN, its Mean Square Error (MSE) is measured. MSE represents the summed square of the difference between the ANN output value and the simulation output value for all samples, according to Equation 2

$$\frac{1}{n}\sum_{j=1}^{n}(m_j - s_j)^2 \qquad (2)$$

where n is the number of samples, m_j is the output from the ANN for sample j, and s_j is the output from the simulation for sample j. Obviously, the lower the MSE value, the better the accuracy of the ANN. MSE is commonly used as the accuracy measurement of ANNs [20].

To prevent from random bias in measuring the MSE, the training of the ANN is replicated 1000 times for each data set. After training, the MSE is measured on a test set of 1000 randomly generated data samples. The average MSE from 1000 replications of training the ANN on the three different data sets is shown in Table 2.

Data set	MSE
Full factorial	0.0132
LH	0.0307
Random	0.0097

Table 2: MSE of ANN for different training sets.

As shown in Table 2, the highest accuracy of the ANN is achieved when training takes place with the randomly sampled data set. One might speculate that this, at least to some degree, can be explained by the fact that the test set used to estimate the MSE is composed of random data. However, it is quite surprising that the results of training with the LH data sets differs that much from training with the random data set, since an LH design involves a high degree of randomness.

In order to get more information about what impact randomness in the training data has on the MSE, two new data sets are composed and used to train the ANN. The first data set is a mix of data from the full factorial and the random data sets, while the second data set is a mix of data from the LH and the random data sets. These new data sets both comprise 243 samples (122 and 121 samples randomly chosen from each of the original data sets, respectively). Table 3 shows the resulting MSE when training the ANN with the new data sets (average from 1000 replications).

Data set	MSE
Full factorial + Random	0.0097
LH + Random	0.0121

Table 3: MSE of ANN trained with mixed training sets.

When comparing the results in Table 2 and Table 3, it is obvious that a better accuracy of the ANN is achieved when more randomness is introduced in the training data set. Once again it is surprising that the LH mixed data set gives the worst ANN, since this data set can be expected to contain the highest degree of randomness.

4.2 OPTIMISATION

The accuracy of the ANN is also evaluated when applied to the optimisation scenario described in Section 3.2. Before the optimisation is started, training of the ANN takes place with the three different data sets (full factorial, LH, and random) using the same procedure as described in Section 4.1. The optimisation is done with the metamodel-assisted evolutionary algorithm described in [21]. In this algorithm, a metamodel is used to identify promising solutions to be evaluated using the time-consuming simulation.

Fig. 7 presents results from the optimisation process (where the goal of the optimisation is to minimise the fitness value). For each of the three data sets, the results presented are the average from three replications of the optimisation. As the chart shows, the best results are achieved when the full factorial data set is used to train the ANN, and worst results are achieved when using the LH data set.

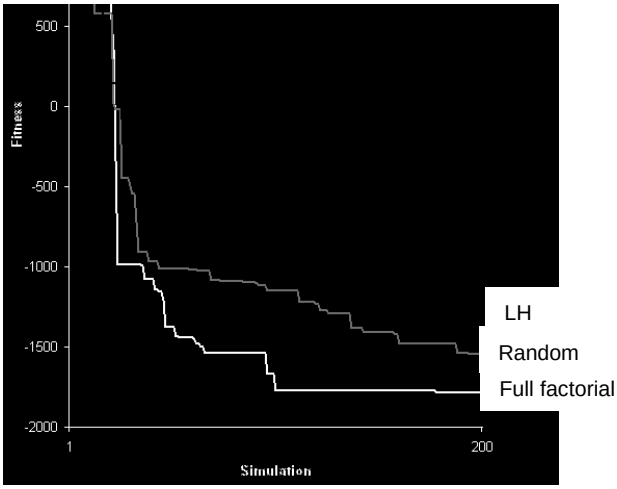

Figure 7: Optimisation results for different data sets.

6. CONCLUSIONS AND FUTURE WORK

In this study, the use of DoE for training metamodels in simulation-based optimisation of manufacturing systems is evaluated. The evaluation is done through a case study of a real manufacturing system. In this case study, an ANN is trained as a metamodel of a simulation model of the real system. Two different DoE designs, full factorial design and LH design, are used for generation of training data for the ANN. In addition, a random training data set is also used for comparison.

The accuracy of the ANN is first evaluated based on MSE values when testing the ANN on previously unseen random data. Results from the experiments show that highest accuracy of the ANN is achieved when training takes place with the randomly sampled data set. Lowest accuracy is achieved with the LH design – the MSE of this design is almost three times worse compared to the MSE of the random data set. The accuracy of the ANN is also evaluated based on its performance when applied in an optimisation scenario. In this case, the experiments show that best results are instead achieved when the ANN is trained with data generated by the full factorial design. Similar to the MSE evaluation, worst results are obtained with data generated using the LH design. Why the results from the MSE experiments and the optimisation experiments do not agree is not clear. One aspect that might partly explain the differences is the number of replications of the optimisation experiments; a larger number of replications can possibly present other results. In the literature, there are also differing statements of whether DoE generated data or random data is to prefer for training of ANNs. Several authors favour DoE ([8], [22]-[23]), but some authors argue that random data is superior [24]. Clearly this question needs to be further investigated.

Regarding future work, this will also focus on evaluating additional DoE techniques. The full factorial and the LH designs are just two out of plenty possible designs, and further investigations is needed to determine which of these designs is the most effective for training of metamodels. For valid results, more test cases with various properties must also be collected and used in the evaluation.

ACKNOWLEDGEMENTS

This work is done within the OPTIMIST project which is partially financed by the Knowledge Foundation (KK Stiftelsen), Sweden. The authors gratefully acknowledge the Knowledge Foundation for the provision of research funding.

REFERENCES

[1] M. Haouani, D. Lefebvre, N. Zerhouni, and A.E. Moudni, "Neural networks implementation for modeling and control design of manufacturing systems", Journal of Intelligent Manufacturing 11, 29-40, 2000.

[2] S. Ólafsson, and J. Kim, "Simulation Optimization", Proceedings of the 2002 Winter Simulation Conference, 79-85, 2002.

[3] J. April, M. Better, F. Glover, and K. Kelly, "New advances for marrying simulation and optimization", Proceedings of the 2004 Winter Simulation Conference, 80-86, Washington, DC, 2004.

[4] A. M. Law, and W. D. Kelton, Simulation Modeling and Analysis, Second Edition, McGraw-Hill, New York, 1991.

[5] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation", Soft Computing 9, 3–12, 2005.

[6] Y.S. Ong, P.B. Nair, A.J. Keane, and K.W. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems", Knowledge Incorporation in Evolutionary Computation, Springer, 307-332, 2004.

[7] K. Mehrotra, C.K. Mohan, S. Ranka, Elements of Artificial Neural Networks, MIT Press, ISBN 0-262-13328-8, 1996.

[8] F. M. Alam, K. R. McNaught, and T. J. Ringrose, "A comparison of experimental designs in the development of a neural network simulation metamodel". Simulation Modelling: Practice Theory 12, 559–578, 2004.

[9] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "A user's guide to the brave new world of simulation experiments", INFORMS Journal on Computing 17, 263–289, 2005.

[10] S.M. Sanchez, (2005) "Work Smarter, Not Harder: Guidelines for Designing Simulation Experiments", Proceedings of 2005 Winter Simulation Conference, 2005.

[11] L. Monostori and Z.J. Viharos, "Hybrid, AI- and simulation-supported optimisation of process chain and production plants", Annals of the CIRP 50(1), 353-356, 2001.

[12] R.W. Blanning, "The construction and implementation of metamodels", Simulation, 177–184, 1975.

[13] P.A. Fishwick, "Neural network models in simulation: a comparison with traditional modeling approaches", Proceedings of the 1989 Winter Simulation Conference, Washington, DC, 702–710, 1989.

[14] H. Pierreval, "Training a neural network by simulation for dispatching problems", Proceedings of the Third Rensselaer International Conference on Computer Integrated Engineering, New York, 332–336, 1992.

[15] H. Pierreval and R.C. Huntsinger, "An investigation on neural network capabilities as simulation metamodels", Proceedings of the 1992 Summer Computer Simulation Conference, San Diego, CA, pp. 413–417, 1992.

[16] I. Sabuncuoğlu and S. Touhami, "Simulation metamodeling with neural networks: an experimental investigation", International Journal of Production Research 40(11), 2483-2505, 2002.

[17] D. J. Fonseca, D.O. Navaresse, and G. P. Moynihan, "Simulation metamodeling through artificial neural networks", Engineering Applications of Artificial Intelligence 16(3), 2003, 177-183.

[18] B. Riedmiller, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA,, pages 586-591, 1993.

[19] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", Proceedings of International Joint Conference on Artificial Intelligence (IJCAI95), 1995.

[20] S. Kang, and W. Kim, "A Novel Approach for Simplifying Neural Networks by Identifying Decoupling Inputs", Australian Conference on Artificial Intelligence 2004, 754-765, 2004.

[21] A. Persson, H. Grimm, H., M. Andersson, and A. Ng, "Metamodel-Assisted Simulation-Based Optimization of a Real-World Manufacturing Problem", Proceedings of The 17th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM'07), Philadelphia, USA, June 18-20, 950-956, 2007.

[22] I. Mezgár, C.S. Eresits, and L. Monostori, "Design and Real-Time Reconfiguration of Robust Manufacturing Systems by Using Design of Experiments and Artificial Neural Network", Computers in Industry 33, 61-70, 1996.

[23] J. P. C. Kleijnen, and R.G. Sargent, "A methodology for fitting and validating metamodels in simulation", The Journal of Operational Research 120, 14-29, 2000.

[24] R.D. Hurrion, and S. Birgil, "A comparison of factorial and random design methods for development of regression and neural network simulation metamodel", The Journal of Operational Research Society 10, 1018-1033, 1999.