Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 16 (2003) 177–183
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Simulation metamodeling through artificial neural networks

D.J. Fonseca*, D.O. Navaresse, G.P. Moynihan

Department of Industrial Engineering, The University of Alabama, Box 870288, Tuscaloosa, AL 35487-0288, USA

Received 10 June 2002; accepted 12 April 2003

Abstract

Simulation metamodeling has been a major research field during the last decade. The main objective has been to provide robust, fast decision support aids to enhance the overall effectiveness of decision-making processes. This paper discusses the importance of simulation metamodeling through artificial neural networks (ANNs), and provides general guidelines for the development of ANN-based simulation metamodels. Such guidelines were successfully applied in the development of two ANNs trained to estimate the manufacturing lead times (MLT) for orders simultaneously processed in a four-machine job shop.

The design of intelligent systems such as ANNs may help to avoid some of the drawbacks of traditional computer simulation. Metamodels offer significant advantages regarding time consumption and simplicity to evaluate multi-criteria situations. Their operation is notoriously fast compared to the time required to operate conventional simulation packages.

Keywords: Artificial neural networks; Metamodeling; Simulation; Job shop sequencing

1. Introduction

Simulation is a widely accepted tool in systems design and analysis. Because its basic concepts are easily understood, it has become a powerful decision-making instrument. It requires a few simplifying assumptions, captures many of the true characteristics of the real model, and provides good insights about the interactions and relationships between qualitative and quantitative variables. However, two major shortcomings of simulation are the need for expert assistance and the significant number of man-hours typically spent in developing the models (Pegden et al., 1995; Schelasin and Mauer, 1995).

Metamodeling techniques have been recently pursued in order to tackle these drawbacks. A metamodel or response surface approximates the usually unknown input–output function implied by the underlying simulation. It attempts to provide a decision-making tool for environments where there are no resources to simulate multiple scenarios. Such situations may occur in numerous settings such as military facilities, hospital emergency departments, fire stations, and manufacturing systems (Kilmer et al., 1997, 1999).

Computer intelligence or artificial intelligence (AI) is the generic name given to a field of computer science dedicated to the development of programs that attempt to be "as smart as humans" in the sense of their abilities to learn. Artificial neural networks (ANNs) is one of the AI techniques that have gained an important role in solving problems with extremely difficult or unknown analytical solution (Lawrence, 1994). ANNs' property of learning from examples makes them powerful programming tools when domain rules are not completely certain or when some amount of inaccuracy or conflicting data exist (Medsker and Liebowitz, 1994).

2. Background

2.1. The ANN paradigm

In its most general form, an ANN can model the way in which the brain performs a particular function. The human brain is a highly complex, non-linear, and parallel information-processing system with the capability of performing certain computations (e.g., pattern recognition, perception, and motor control) many times

*Corresponding author. Tel.: +1-205-348-3337; fax: +1-205-348-7162.
E-mail address: dfonseca@coe.eng.ua.edu (D.J. Fonseca).

0952-1976/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0952-1976(03)00043-5

faster than the fastest digital computer in existence today. ANNs are simulated neurons interconnected in similar manner as the human brain's neurons.

There are three basic elements of an ANN model, which are illustrated in Fig. 1 (Haykin, 1999): (a) a set of connecting links or *synapses*, each characterized by a *weight* of its own. Specifically, an input signal to the synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}, (b) an *adder*, which sums the input signals weighted by the respective synapses of the neuron, and (c) an *activation function,* which limits the permissible amplitude range of the output signal to some finite value. It defines the output of the neuron in terms of the induced local field, which is formed by the *linear combiner output* u_k, and the *bias* b_k. This externally applied *bias* is used to increase or lower the net input of the activation function.

The ability to learn and improve its performance from examples is the ANN's fundamental trait. Haykin (1999) defines as learning, in the ANNs context, the process by which the free parameters of the ANN are adapted through stimulation provided by the environment where the network is embedded. Instead of following a set of rules, ANNs are able to learn underlying relationships from a collection of training examples. ANNs are usually classified into two main categories: *recurrent* networks, in which loops occur because of feedback connections, and *feed-forward* networks, in which the network structure has no loops. The choices of network architectures are intimately linked to the learning algorithm used in the training of the network.

Multilayered, feed-forward, non-linear network models are utilized for general-purpose and generalization applications. These types of networks are commonly known as Multilayer Perceptions (Maps). Maps have been successfully applied to solve diverse problems by training them in a supervised environment. The back error propagation algorithm (BEEP) presented by Rumelhart et al. (1996), also known as the *Generalized Delta Rule,* is the most widely used supervised learning algorithm for MAPS. BEEP learns to generate a mapping from the input space to the output space by minimizing the error between the desired output and the actual output produced by the network. At each iteration, neurons slightly adjust their input connection weights in the direction that reduces their signal errors. This process is repeated for subsequent training patterns. Once trained, the MLP is able to identify features in input patterns, and to produce meaningful outputs based on the detected presence (or absence) of those features in a new pattern. In short, BEEP is a *gradient-descent* technique, which is basically an iterative version of the simple Least-Squares Method (LSM), adapted to non-linear, multidimensional relationships.

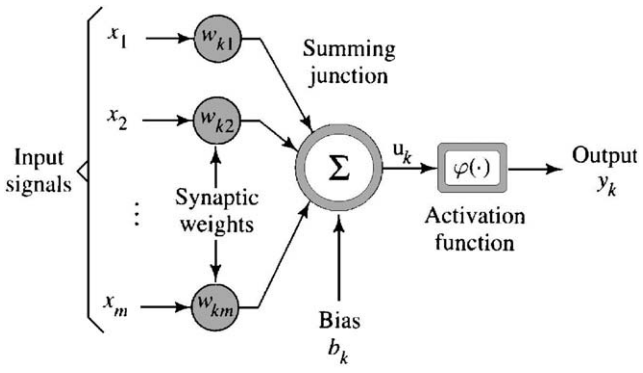

Fig. 1. Non-linear model of a neuron.

2.2. Simulation metamodeling through ANNs

Systems simulation can be applied to virtually any type of problem, and it is frequently used when quantitative methods fail in providing acceptable solutions. It is known that simulation does not provide a solution; but rather represents a valuable analytical tool, especially in situations when determining factors are stochastic in nature.

The main goals of simulation metamodels are to reduce the cost, time, and amount of effort required by a simulation analysis. A metamodel, or response surface, is an approximation of the input/output function implied by the underlying simulation model. It is usually a supplementary model that can be alternatively used to interpret a more detailed model.

Regression has been for more than two decades the most popular method to perform metamodeling of simulations. However, the almost simultaneously published works of Hornik et al. (1989) and Funahashi (1989) rigorously demonstrated that Multilayer ANNs were theoretically capable of approximating any measurable function to any desired degree of accuracy. They provided a fundamental basis to establish that these networks were "general universal approximators". Pierreval (1992) and Pierreval and Huntsinger (1992) then used neural networks to model the simulation of manufacturing shops. First, Pierreval proposed a neural network approach to the selection of dispatching rules on a simplified flow shop. The network was trained with data obtained from simulation and was able to select for itself the best dispatching rule for new cases. Then, he and Huntsinger applied ANN metamodeling capabilities to a discrete system: a job shop with jobs arriving at a constant rate, and with deterministic processing and transportation times. Both of these studies yielded results that encouraged the use of ANNs as simulation metamodels, particularly where computing costs were important.

In addition, Badiru and Siege (1998) implemented an ANN-based simulation metamodel to predict future cash flow values for cost estimation, using the initial

investment and the interest rate per time period as uniform random variables. Finally, Kilmer et al. (1997) described the use of supervised neural networks as a metamodeling technique for discrete, stochastic simulation. The first work provided the foundations for developing ANN metamodels. The second and third ones reported the development of ANN metamodels for the simulation of a hospital emergency department, and a manufacturing warehouse, respectively.

The reported studies are evidence that metamodeling through ANNs may overcome many of the disadvantages of traditional regression approaches. Regression is limited to approximate a subset of the simulation domain, is sensitive to deviations from statistical model assumptions such as constant error variance, and there exists the possibility of selecting an incorrect functional form. ANN-based models can achieve a global fit, remove the possibility of pre-selecting improper approximation functions, are able to accommodate together discrete and continuous variables, are insensitive to common deviations from statistical model assumptions, and are robust to incomplete, noisy, or inaccurate data (Kilmer et al., 1997).

3. Guidelines for the development of ANN stochastic simulation metamodels

Several guidelines and recommendations to approach the development of ANNs are found in the literature: Kilmer et al. (1999), Skapura (1996), Lawrence (1994), Crooks (1992), Pierreval and Huntsinger (1992), Caudille (1991), and Lawrence (1991). Even though most studies suggest certain methodologies for MAPS development using BEEP, finding a net architecture capable of good generalization is generally reported as a trial-and-error process. Some steps for developing a simulation metamodel through ANNs are presented here.

Step 1: *Simulation model development*. This step consists of creating the simulation model to be used as the baseline for the ANN metamodel. It involves the selection of relevant input and response variables, the construction of the basic simulation model for generating the training and testing facts, and the determination of the appropriate number of replications for each developed scenario in to ensure statistical confidence.

Step 2: *Data preparation*. For practical applications, input data must satisfy some necessary requisites. One condition is that the input subset contains enough information for the ANN to devise the target response. The experimenter cannot expect an ANN to learn a non-existent function. A second condition requires that the function to be learned be *smooth*. This means that a small change in the inputs should mostly create a small change in the response. Extremely *non-smooth* functions cannot be successfully learned by ANNs.

After completion of the facts generation step, data must be pre-processed before feeding the ANNs. The range of the data needs to be limited. This is usually accomplished by normalizing inputs and outputs to values between 0 and 1 or −1 and 1. The order in which facts are presented to the ANN should be randomized, since ANNs can have trouble generalizing if the training samples are grouped by similar inputs or outputs (Lawrence, 1994).

Step 3: *ANN development*. Definition of an ANN architecture means determining the number of input, output, and hidden neurons, and the number of hidden layers (HL). The *Universal Approximation Theorem* states that an MLP with only one HL is sufficient to compute an approximation of a given function by using a given training set. However, this does not mean that a single HL is optimum regarding generalization, learning time, and ease of implementation (Haykin, 1999). Perhaps the most practical approach regarding the number of HLs is to start experimentation with one HL and then, if training fails, increase the number of HLs.

Once the number of HL is decided, the number of hidden neurons per layer—and consequently, the number of connections in the networks—is determined based on both the number of training facts and the number of input and output neurons. This is important, because of the well-documented risk associated with having an excessive number of connections: the network may end up memorizing the facts rather than learning to generalize about them. Such a phenomenon is known as *overfitting* and occurs when the network possesses too many degrees of freedom (synaptic weights) compared to the number of available training facts. If this happens, the network may train well, but its testing is deficient. While some authors recommend that the number of training facts should not exceed ten times the number of connections, others suggest to start the training with a small number of hidden neurons, and then to add one hidden neuron every time the training stops progressing (Lawrence, 1994; Haykin, 1999; Skapura, 1996).

Step 4: *Models' training and testing for system selection*. In some instances, an ANN may test well before properly assimilating all given training facts. It must then be kept in mind that a network that performs well on new data is frequently much better than a network that is able to learn the 100% of the training data. Since generalization is the fundamental trait pursued during an ANN design, a "testing while training" approach may help to find ANNs that generalize well earlier. In this way, after each run over the whole training set, a previously separated set of facts is executed, and the number of "good" and "bad" facts is computed and saved. The current network architectures are periodically saved during training in order to

choose, at the end of the training, the "best-testing network".

The conditions imposed by the *Universal Approximation Theorem* regarding transfer functions, i.e., non-constant, bounded, and monotone-increasing functions, are satisfied by sigmoidal functions. Popular examples are the logistic and the hyperbolic tangent, widely used for training MAPS by BEEP.

There are other instruments that the designer must be able to handle for training control. For instance, tolerance tuning during training to reduce error by a given factor any time the percentage of good facts reaches a high value may help convergence. A low reduction limit (0.01 or less) can ensure that the training would not terminate prior to reaching the maximum number of runs. Momentum is a variable that adds a delta to the error correction, which is in the direction of the last run's overall error adjustment. This helps the network move in only one direction. Using a momentum of zero would imply that the past direction of the error is irrelevant, what might considerably slow the training process.

After completion of a specified number of training runs, charts depicting the number of good facts and/or some type of error measure (root mean squared error—RMS, or mean average error—MAE) as a function of executed epochs can be generated. Inspection of these charts makes possible to select networks with the smallest testing error.

Step 5: *Networks improvement*. Once a small group of "good-testing" networks are selected, some extra experiments can be carried out in order to improve their performances. These can include the addition of small amount of noise to the training data, the pruning of small connections, or some trials on networks with a higher number of HL (Lawrence, 1994).

Step 6: *ANN validation*. Validation of the selected ANN models is a critical phase in any study. Validation involves determining whether the conceptual model is an accurate representation of the real process. The selected ANN models can be validated by executing a set of simulation situations different from those used in the training and testing phases. A practical procedure to validate situations where correct answers are unknown, as it occurs in numerous simulation studies, involves comparing the results of the ANN models with the results from a certain number of different simulation packages.

4. Metamodeling of a job shop sequencing simulation: a case study

The discussed procedure was implemented by using a four-machines/four-job situation as problem domain. To accomplish this, a problem from Askin and Standridge (1993) was extracted, enhanced, and a degree of uncertainty was added to the processing times. The considered job shop involved a set of four machines, e.g. A, B, C, and D, used to carry out independent manufacturing operations. Thus, an order that required processing by machines A, C, and D could follow any possible machine sequence, e.g., ABC, BCA, CBA, and so on (i.e., precedence constraint relaxed). Different sequences for each job type yielded different average manufacturing lead times (MLT). The expectation was that sequencing decisions based on ANN-based simulation would exhibit a similar degree of validity than decisions based on traditional simulation modeling. The simulation software package Arena was used to create the set of examples necessary to train and test the prototype networks. Feed forward, multi-layered, fully connected MAPS trained through BEEP were studied. BrainMaker, a commercial ANN simulation software package, was chosen as the development shell.

After numerous trials following the described guidelines, the two best performing networks were selected. At the end of each training process, charts such as the ones illustrated in Figs. 2 and 3 were constructed. They showed the average testing error, the number of good testing facts, and the testing RMS error for every testing run. Only those epochs with correct training facts percentages higher than 75% were tested and plotted. Through the information inferred from these charts, and other relevant collected testing statistics, the runs with minimum RMS values were identified. This same procedure was systematically applied to select the best network of the SCS that succeeded to complete at least one training process.

Three independent validation procedures were performed. A set of 100 validation scenarios was executed through Arena, SIMAN, ProModel, and the two selected ANNs. The first one consisted of a statistical test to compare the average values yielded by each model. Second, a criterion for classifying the results obtained from Arena (the software used to create the training set), the ANNs, SIMAN, and ProModel was developed. Results from these four models were compared to Arena's, and were classified as "identical", "acceptable", or "different" based on predefined criteria. Finally, a face validation was carried out by plotting the flowtimes yielded by the ANNs against their Arena's counterparts. Statistical analysis performed through Duncan's comparison tests showed evident similarities in the averages between Arena's and ANNs', and between Arena's and SIMAN's estimations. The classification procedure based on similarity of results reinforced the conclusions of that analysis. The results obtained from the ANN-based simulations and the engine that generated the training data, i.e., Arena, were also graphically compared by plotting the ANN's estimations superimposed on a 5% Arena's tolerance

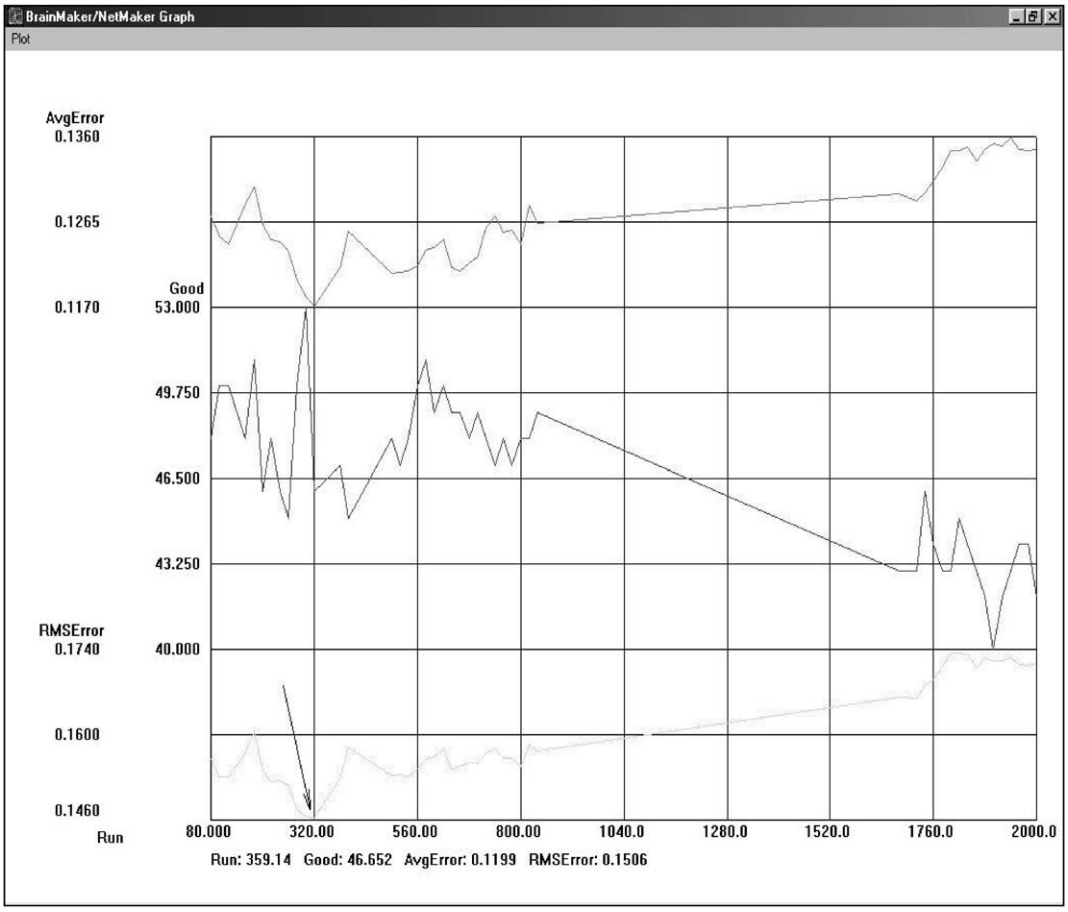

Fig. 2. Testing statistics plot.

interval (Fig. 4). The same procedure was used for SIMAN and ProModel. Again, the ANN's and SIMAN's results seemed to follow the trend insinuated by Arena. ProModel's results were distant from Arena's estimations. The two developed models were able to fairly capture the complex relationship between machine sequences and resulting flowtimes variation.

5. Conclusions

The main objective of this research was to contribute to the understanding of the mechanics involved in designing ANNs for simulation metamodeling. The appropriate design of intelligent systems such as ANNs may help to avoid some of the drawbacks of traditional computer simulation. A case study based on discrete-event stochastic simulation was used to create the training and testing sets to feed the ANNs. Machine sequences for each different job type constituted the decision variables, while orders' average flowtimes, the outputs.

ANNs proved to be a viable tool for stochastic simulation metamodeling. Despite the noise introduced by the stochastic nature of most simulation processes, the ANN-based simulations may be able to fairly capture underlying relationships between appropriately selected inputs and responses.

The development of adequate codification for the input variables may become a critical issue in the ANNs design. For the presented case study, the number of necessary digits for representing a given sequence greatly affected the number of connections between the input and HL, and the ANNs capabilities of effectively learning were highly restricted by the selected codification scheme.

From the practical application standpoint, metamodels may offer significant advantages regarding time consumption and simplicity to evaluate new job shop situations. The software operation is notoriously fast compared to the time required to operate any conventional simulation software. This feature can become more and more relevant as the model is expanded and enlarged. In addition, ANN models properly embedded in adequate DSS, may greatly contribute to the simplification of decision-making processes in areas where there are no resources to conduct a simulation study, or when a fast response is required.

NetMaker Professional - TestStat-800-26-30.sta

File Column Row Label Number Symbol Operate Indicators

	Run	TotFacts	Good	Bad	adOutput	TotalBad	Learn	Tolerance	AvgError	RMSErro	h:mm:ss	Ry:F1	Ry:F2	Ry:F3	Ry:F4
1	80	80	48	32	69	69	0.3174	0.2	0.1271	0.156	00:01:11	0.0033	0.0017	0.3736	0.2533
2	100	160	50	30	65	134	0.3034	0.2	0.1249	0.1532	00:01:22	0.0102	0.0025	0.3694	0.2504
3	120	240	50	30	67	201	0.31	0.2	0.1241	0.1532	00:01:34	0.0134	0.0036	0.3649	0.2628
4	160	320	48	32	69	270	0.3162	0.2	0.1284	0.157	00:01:53	0.0233	0.0044	0.3775	0.2344
5	180	400	51	29	67	337	0.311	0.2	0.1304	0.1606	00:02:03	0.0231	0.0029	0.399	0.2415
6	200	480	46	34	67	404	0.2884	0.2	0.1263	0.154	00:02:12	0.0263	0.0033	0.3981	0.2448
7	220	560	48	32	57	461	0.2584	0.2	0.1246	0.1523	00:02:22	0.0321	0.0042	0.4271	0.237
8	240	640	46	34	61	522	0.2708	0.2	0.1243	0.1525	00:02:31	0.0325	0.0056	0.4285	0.2229
9	260	720	45	35	57	579	0.256	0.2	0.1232	0.1517	00:02:41	0.0307	0.0066	0.43	0.245
10	280	800	50	30	55	634	0.2672	0.2	0.1201	0.1477	00:02:50	0.0325	0.0095	0.4317	0.2388
11	300	880	53	27	48	682	0.2744	0.2	0.1182	0.1465	00:02:59	0.0304	0.0077	0.451	0.2509
12	320	960	46	34	56	738	0.2622	0.2	0.1171	0.1464	00:03:09	0.0209	0.0064	0.4374	0.2767
13	380	1040	47	33	62	800	0.3234	0.2	0.1214	0.1529	00:03:47	0.0173	0.0114	0.4737	0.2973
14	400	1120	45	35	62	862	0.3224	0.2	0.1255	0.158	00:03:56	0.0141	0.0121	0.4758	0.2752
15	500	1200	48	32	63	925	0.2984	0.2	0.1208	0.1534	00:04:46	0.0133	0.0079	0.4081	0.2547
16	520	1280	47	33	62	987	0.2646	0.2	0.1209	0.1535	00:04:55	0.0102	0.0076	0.4097	0.2586
17	540	1360	48	32	61	1048	0.2622	0.2	0.1211	0.1532	00:05:05	0.0105	0.0058	0.4114	0.2375
18	560	1440	50	30	61	1109	0.2884	0.2	0.1217	0.1545	00:05:14	0.0066	0.004	0.3981	0.2464
19	580	1520	51	29	58	1167	0.2846	0.2	0.1234	0.1559	00:05:22	0.0045	0.0031	0.412	0.2303
20	600	1600	49	31	61	1228	0.2772	0.2	0.1237	0.1563	00:05:31	0.0068	0.003	0.4044	0.2032
21	620	1680	50	30	63	1291	0.296	0.2	0.1245	0.1578	00:05:40	0.0053	0.0043	0.4025	0.1958
22	640	1760	49	31	62	1353	0.2846	0.2	0.1214	0.1542	00:05:48	0.0034	0.0053	0.407	0.2266
23	660	1840	49	31	63	1416	0.27	0.2	0.121	0.1548	00:05:57	0.0027	0.0031	0.4156	0.2305
24	680	1920	48	32	64	1480	0.2672	0.2	0.1219	0.1554	00:06:05	0.0019	0.0048	0.3983	0.2225
25	700	2000	49	31	61	1541	0.2584	0.2	0.1226	0.1555	00:06:14	0.0005	0.0054	0.3748	0.2317
26	720	2080	48	32	62	1603	0.2622	0.2	0.1258	0.157	00:06:22	0.0007	0.0054	0.3652	0.2435
27	740	2160	47	33	62	1665	0.25	0.2	0.1272	0.1578	00:06:31	0.0025	0.0066	0.3376	0.23

Fig. 3. Testing statistics.

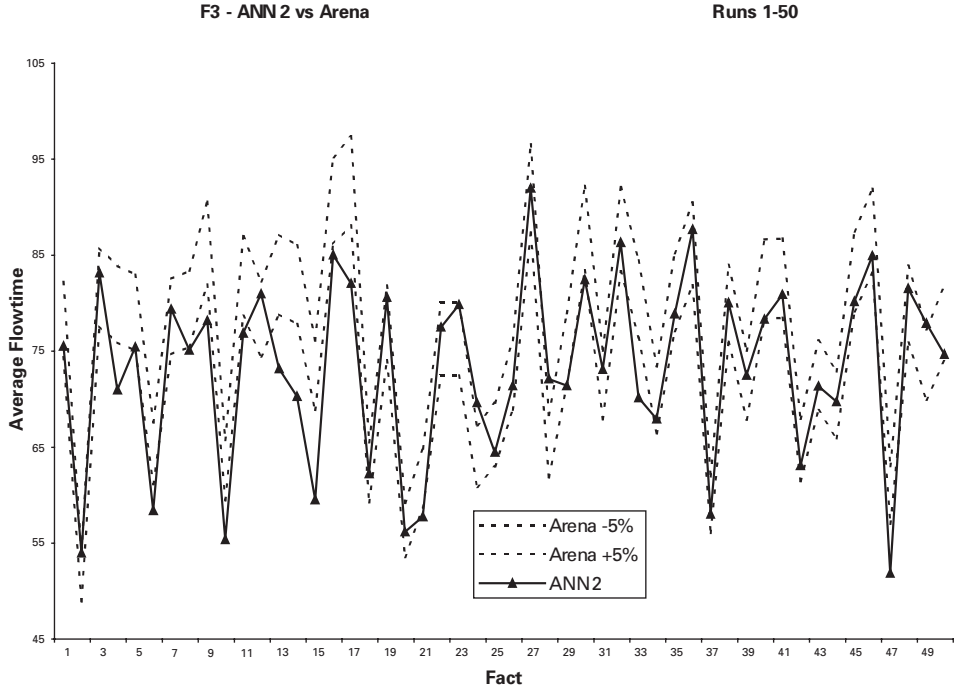

Fig. 4. Job type 3's Flowtimes—ANN 2 vs. Arena (scenarios 1–50).

References

Askin, R., Standridge, C., 1993. Modeling and Analysis of Manufacturing Systems. Wiley, New York, NY.

Badiru, A., Siege, D., 1998. Neural network as a simulation metamodel in economic analysis of risky projects. European Journal of Operational Research 105, 130–142.

Caudille, M., 1991. Neural network training tips and techniques. AI Expert 11, 56–61.

Crooks, T., 1992. Care and feeding of neural networks. AI Expert 12, 36–49.

Funahashi, K., 1989. On the approximate realization of continuous mappings by neural networks. Neural Networks 2, 183–192.

Haykin, S., 1999. Neural Networks. Prentice-Hall, New Jersey, NY.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366.

Kilmer, R., Smith, A., Shuman, L., 1997. An emergency department simulation and a neural network metamodel. Journal of the Society for Health Systems 5 (3), 63–79.

Kilmer, R., Smith, A., Shuman, L., 1999. Computing confidence intervals for stochastic simulation using neural network metamodels. Computers and Industrial Engineering 36, 391–407.

Lawrence, J., 1991. Data preparation for a neural network. AI Expert 11, 34–41.

Lawrence, J., 1994. Introduction to Neural Networks: Design, Theory, and Applications. California Scientific Software, Nevada City, CA.

Medsker, L., Liebowitz, J., 1994. Design and Development of Expert Systems and Neural Networks. MacMillan College Publishing, New York, NY.

Pegden, C., Shannon, R., Sadowsky, R., 1995. Introduction to Simulation using SIMAN. McGraw Hill, Blacklick, OH.

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

Pierreval, H., Huntsinger, R., 1992. An investigation on neural network capabilities as simulation metamodels. In: Proceedings of the 1992 Summer Computer Simulation Conference, California, CA, pp. 413–417.

Rumelhart, D., Hinton, G., Williams, R., 1996. Learning representations by error propagation. Parallel Distributed Processing 1, 318–362.

Schelasin, R., Mauer, J., 1995. Creating flexible simulation models. IEEE Solutions 5, 50–67.

Skapura, D., 1996. Building Neural Networks. Addison-Wesley Publishing, New York, NY.

Daniel J. Fonseca is an Assistant Professor of Industrial Engineering at The University of Alabama. Prior to joining the faculty of The University of Alabama, Dr. Fonseca worked as a professor and industrial consultant for the Monterrey Institute of Technology in Mexico. He served as an engineering consultant in the design and improvement of manufacturing and service systems in medium to large-scale companies in Mexico City and Tampico. Dr. Fonseca received B.S. and M.S. degrees in Industrial Engineering from The University of Alabama, and M.S. and Ph.D. degrees in Engineering Science from Louisiana State University. His research interests include systems simulation, production management, and artificial intelligence.

Daniel O. Navaresse is currently working as a strategic planner for American Beverages in San Paulo, Brazil. Prior to joining American Beverages, Mr. Navaresse worked as a service analyst for Varig Airlines, and as a process engineering for Tetra Pack Inc. Mr. Navaresse has a B.S. degree in Aerospace Engineering from Universidad Tecnologica Nacional of Argentina, and an M.S. degree in Industrial Engineering from the University of Alabama. His research interests include operation management, and systems simulation.

Gary P. Moynihan is a Professor of Industrial Engineering at The University of Alabama. Dr. Moynihan has held positions in the aerospace, computer, and chemical processing industries. He was employed in production supervision and manufacturing management by the American Cyanamid Corporation and National Micronetics, and by Martin Marietta Aerospace. Dr. Moynihan received a B.S. degree in Chemistry, and an MBA degree from the Rensselaer Polytechnic Institute, and a Ph.D. degree in Industrial Engineering from the University of Central Florida. His research interests include management information systems, and engineering management.