

**SIMULATION METAMODELING WITH NEURAL
NETWORKS**

A THESIS

**SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL
ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE**

By

Souheyl Touhami

June, 1997

TS
155.63
T68
1997

SIMULATION METAMODELING WITH NEURAL NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL
ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By

Souheyl Touhami

June, 1997

To
1996.03
TGS
1997

8037974

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



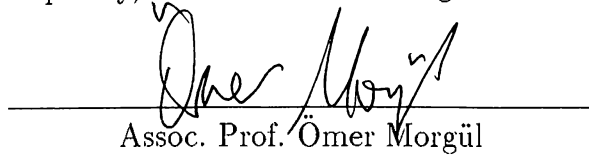
Assoc. Prof. İhsan Sabuncuoğlu (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Osman Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Ömer Morgül

Approved for the Institute of Engineering and Sciences:



Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

SIMULATION METAMODELING WITH NEURAL NETWORKS

Souheyl Touhami

M.S. in Industrial Engineering

Supervisor: Assoc. Prof. İhsan Sabuncuoğlu

June, 1997

Modern manufacturing environments increasingly call for more sophisticated and fast decision aiding systems for their management. Artificial neural networks have been proposed as an alternative approach for formalizing various quantitative and qualitative aspects of manufacturing systems. This research attempts to lay down the motivation behind using neural networks as a simulation metamodeling approach. This research can be classified under the major headings of simulation metamodeling for the purpose of estimating system performance. Steady state performance of non-terminating type systems and transient state performance of terminating type systems are examined under job shop environments by applying Back Propagation neural networks. We attempt to study the performance of neural metamodels with respect to estimating two performance measures (mean machine utilization and mean job tardiness), with respect to system complexity, with different types of system configurations (deterministic and stochastic), with respect to multiple metamodel accuracy assessment criteria and various metamodel design settings. The objective of this analysis is to investigate the potential application of neural metamodeling.

Key words: Simulation, Metamodeling and Neural Networks.

ÖZET

YAPAY SİNİR AĞLARI İLE BENZETİM META MODELLERİNİN OLUŞTURULMASI

Souheyl Touhami

Endüstri Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Dr. İhsan Sabuncuoğlu

Haziran, 1997

Günümüzde modern imalat sistemleri daha karışık ve hızlı karar veren yöntemlere ihtiyaç duymaktadır. Bu amaca yönelik olarak, yapay sinir ağları alternatif yöntem olarak önerilmektedir. Bu çalışmada, yapay sinir ağlarının bu tür yöntemlerde kullanılmasını sağlayacak temeller oluşturulmaktadır. Gerek uzun dönemli ve gerekse kısa vadeli sistem performansını ölçecek modeller oluşturulmaktadır. Geri yaymalı (back propagation) yöntemine dayalı olarak geliştirilen yapay sinir ağları sistemin ortalama kullanım oranı ve artı gecikme zamanı performans ölçütlerini tahmin etmekte kullanılacaktır. Önerilen yaklaşımlar ve geliştirilen modellerin başarısı çeşitli sistem koşullarında farklı değerlendirme kriterine göre ölçülecektir.

Anahtar sözcükler: Benzetim, Meta modellemesi, Yapay Sinir Ağları

to my parents, to my sisters

ACKNOWLEDGEMENT

I would like to express my deep gratitude to Dr. Ihsan Sabuncuoğlu for his guidance, attention, understanding and patience throughout all this work.

I am indebted to the readers Dr. Osman Oğuz and Dr. Ömer Morgül for their effort, kindness, and time.

I cannot fully express my gratitude and thanks to my friends for their care, support and encouragement.

Souheyl Touhami.

Contents

1	Introduction	1
2	Research Background	4
2.1	Simulation	4
2.2	Simulation Metamodels	7
2.3	Neural Networks	10
2.4	Neural networks as a simulation metamodeling approach	19
3	Estimating Long Term Machine Utilization	25
3.1	Case 1: Simple System	26
3.1.1	Experimental settings	26
3.1.2	Results and Discussions	33
3.2	Case 2: Complex System	36
3.2.1	Experimental settings	37
3.2.2	Results and Discussions	40
3.3	Comparison of Simple vs. Complex systems	42

<i>CONTENTS</i>	viii
4 Estimating Long Term Job Tardiness	44
4.1 Case 1: Simple System	45
4.1.1 Experimental settings	45
4.1.2 Results and Discussions	51
4.2 Case 2: Complex System	55
4.2.1 Experimental settings	55
4.2.2 Results and Discussions	57
4.3 Comparison of Simple & Complex systems	59
5 Estimating Short Term Job Tardiness	62
5.1 Experimental settings	63
5.2 Results and Discussions	66
6 CONCLUDING REMARKS	74
A Figures	85
B Tables	98
C Codes	134

List of Figures

2.1	Metamodeling Concept.	9
2.2	The generic network architecture used.	12
2.3	Single processing unit in neural network.	13
3.1	Mean Utilization: Simple System: Relationship between models.	28
3.2	Mean Utilization and Mean Tardiness: Complex System: Relationship between models.	38
4.1	Mean Tardiness: Simple System: Relationship between models. .	47
5.1	Mean Tardiness: Simple System: Short term performance estimation: Relationship between Metamodels.	65
3.1.2	Mean Utilization: Metamodel performance through assessment criteria.	86
3.1.3	Learning curve for selected best neural network.	87
3.2.2	Mean Utilization: Complex System: Metamodel performance through assessment criteria.	88
3.3	Mean Utilization: Comparisons.	89

4.1.2 Mean Tardiness: Simple System: Metamodel performance through assessment criteria.	90
4.2.1 Mean Tardiness: Complex System: Metamodel performance through assessment criteria.	92
4.3 Mean Tardiness: Comparisons.	94
5.2 Mean Tardiness: Simple System: Short term performance estimation: Metamodel performance through assessment criteria.	95
5.3 Mean Tardiness: Short term performance estimation: Comparing effect of stochasticity and system complexity.	97
2.1 Metamodeling Concept.	9
2.2 The generic network architecture used.	12
2.3 Single processing unit in neural network.	13
3.1.1 Mean Utilization: Simple System: Relationship between models.	28
3.1.2 Mean Utilization: Metamodel performance through assessment criteria.	28
3.1.3 Learning curve for selected best neural network.	28
3.2.1 Mean Utilization and Mean Tardiness: Complex System: Relationship between models.	38
3.2.2 Mean Utilization: Complex System: Metamodel performance through assessment criteria.	28
3.3 Mean Utilization: Comparisons.	28
4.1.1 Mean Tardiness: Simple System: Relationship between models. .	47

4.1.2 Mean Tardiness: Simple System: Metamodel performance through assessment criteria.	47
4.2.1 Mean Tardiness: Complex System: Metamodel performance through assessment criteria.	47
4.3 Mean Tardiness: Comparisons.	47
5.1 Mean Tardiness: Simple System: Short term performance estimation: Relationship between Metamodels.	65
5.2 Mean Tardiness: Simple System: Short term performance estimation: Metamodel performance through assessment criteria.	65
5.3 Mean Tardiness: Short term performance estimation: Comparing effect of stochasticity and system complexity.	65

•

List of Tables

3.1.1 Mean Utilization: Simple System: List of models.	99
3.1.2 Mean Utilization: Simple System: List of neural metamodels. . .	99
3.1.3 Mean Utilization: Simple System: Results of metamodels. . . .	100
3.1.4 Error report of Pierreval's network.	102
3.1.5 Robustness test.	102
3.2.1 Mean Utilization: Complex System: List of models.	103
3.2.2 Mean Utilization: Complex System: Generated set characteristics.	103
3.2.3 Mean Utilization: Complex System: List of neural metamodels.	104
3.2.4 Mean Utilization: Complex System: Results of metamodels. . .	105
4.1.1 Mean Tardiness: Simple System: List of models.	109
4.1.2 Mean Tardiness: Simple System: Generated set characteristics. .	109
4.1.3 Mean Tardiness: Simple System: List of neural metamodels. . .	110
4.1.4 Mean Tardiness: Simple System: Results of metamodels.	112
4.2.1 Mean Tardiness: Complex System: List of models.	120

4.2.2 Mean Tardiness: Complex System: Generated set characteristics.	120
4.2.3 Mean Tardiness: Complex System: List of neural metamodels. .	121
4.2.4 Mean Tardiness: Complex System: Results of metamodels. . . .	122
5.1 Mean Tardiness: Short term estimation: List of models.	126
5.2 Mean Tardiness: Short term estimation: Generated set characteristics.	127
5.3 Mean Tardiness: Short term estimation: Results of metamodels.	128

Chapter 1

Introduction

Simulation has been widely accepted by the OR community and the business sector as a valuable tool in solving large problem instances that are unsolvable (or expensively solvable) with other quantitative approaches. However, due to the time requirements and lack of optimization capabilities, simulation may not be appropriate for real time applications, which are more and more calling for faster techniques. The use of simulation metamodels may help solve such problems. Research in metamodeling is maturing. Since 1987, a resurgence of interest mostly appears as case studies. This is the case also for use of neural networks as a metamodeling approach, which are quite recent. The case studies reported in the literature are not elaborated enough to allow assessing their potential applications in real life.

The aim of this work is to investigate the boundaries of simulation metamodeling with Artificial Neural Networks for the purpose of estimating system performance measures in job shop environments. This study is based on neural networks that operate with the Back Propagation algorithm. This research has two major parts. In the first part, we evaluate the performance of the neural networks in estimating long term or steady state performance of non-terminating type simulations. We attempt to determine the effect of system performance measures (mean job tardiness vs. mean machine utilization), system configuration (deterministic vs. stochastic), system complexity (simple

vs. complex), error assessment criteria and network design settings on the predictive capabilities of the designed neural metamodels. In the second part, we evaluate the neural networks in estimating short term or transient state system performance with terminating type simulation. In this latter part, the initial system status plays an important role. For this we investigate the effect of the initial system status, demand on system, error assessment criteria and network design settings on the predictive capabilities of the designed neural metamodels. A simulation investigation of a particular system might be either terminating or non-terminating, depending on the objectives of the study. The terminating type simulation is one for which there is a “natural” event that specifies the length of the simulation run and the nonterminating type is the one for which there is no such event. In our experiments, we assume that the objective of the study is to evaluate the long term and short term impact of the selected operational policies and hence we assume that the ending event for the terminating simulation is imposed by the management and is specified in terms of time. For the non-terminating simulation, simulation run lengths are set as to reveal steady state system behavior.

It has not been of primary emphasis for us to find the best (most precise) neural network metamodel for each of the systems that are studied. Therefore, all the results achieved could be improved through further fine tuning of the experiments. However, we believe that these improvements will not alter the conclusions inferred from this work. The results achieved in the experiments show that neural networks are very promising tools for estimating steady state system performance. For estimating short term system performance, the experiments show that it is a more difficult task and we state some of the factors that influence the performance of neural networks. The experiments indicate that, although neural networks are promising, application to real life may not be straight forward as the existing literature may lead us to expect.

This manuscript consists of 6 chapters. The next chapter lays down the background of our research. The next two chapters are under the major heading of simulation metamodeling with neural networks of non-terminating type systems. The third chapter is related to predicting mean machine

utilization. The fourth chapter is related to predicting mean job tardiness. The fifth chapter reports the work done on simulation metamodeling with neural networks of terminating type systems. In the last chapter we give our conclusions and future research directions. All the related tables, figures and graphs related to this work are provided in the Appendices.

Chapter 2

Research Background

2.1 Simulation

Considering the inherent complexities in modern manufacturing, it is of prime importance for modern management to quickly evaluate the impact of their operational policies on the overall short term and longer term performance of the system before actual implementation takes place, in order to keep up with the dynamic nature of modern business. Thus, the analysis tool used must be fast and with an acceptable degree of precision [12]. When analytical methods can be employed, they can generate the best model of a system. However, due to the strict assumptions required on system states and the complex and lengthy mathematical derivations involved, many analytical models cannot be applied to large or complex systems. Computer simulation is frequently used in these circumstances as an alternative solution approach to solve such problems.

Simulation is a key decision making tool in an advanced manufacturing environment. It reduces the cost, time and risks compared to experimenting decision alternatives with real systems in real time. Simulation allows evaluating short term and long term effect of decision made at all the levels of the manufacturing system. Either used at the system design phase or when operating the system, simulation is a flexible system analysis tool that

allows modeling relatively large systems without requiring many restrictive assumptions. It is used where other approaches find it difficult in terms of modeling and computational requirements [31]. Thus, it is a complementary tool and not in competition with other approaches. Simulation is applicable at all the levels of the hierarchical decision making process, allowing to perform sensitivity analysis and to evaluate different policies at different degrees of aggregation under selected experimental conditions.

On the other hand, the use of simulation has its drawbacks too. Despite all what has been and is being done now, and despite the attention paid to experimental design techniques in order to enhance the value of simulation, practitioners still face some major problems in using it. Simulation is still time and computer memory consuming both when constructing the models or when using them. Moreover, simulation is by its nature a trial and error process. Hence, simulation is mainly used to answer *What-if* questions and so it is useful as an aid to the controller and not as a controller by itself since it is unable to provide best solution directly. As a result, it requires time, skill and experience for a proper analysis and interpretation of simulation results.

From the current practice, simulation applications can be classified into a) stand alone applications and b) hybrid applications [16]. In the first case, simulation models are used to evaluate different design alternatives and/or operational policies without disturbing the actual system. The aim of such applications is in general related to get the overall picture about the system and hence they are more related to the long term impact of decisions. For the hybrid applications, simulation is combined with other tools such as expert systems [28]/artificial intelligence and analytical tools [16][34]. Such hybrid applications are often applied for real time decision making and control of manufacturing systems. Real time scheduling has been approached by other methods. Harmonosky and Robohn [11] present a review of some of these applications, among which simulation and simulation combined with artificial intelligence are reported to play a major role in decision support systems for real time control and scheduling.

When it comes to real time control, the choice of the tool to be used is constrained -among others- by time requirements and precision. Harmonosky and Robohn [12] present an initial investigation of the application potential of simulation to real time control decisions in terms of CPU requirements. Their work shows that CPU requirements is very much dependent on the system being modeled and on the objectives of the application. Thus, time requirement are a major issue that may reduce the application potential of simulation for the control of the manufacturing environment. This fact is more highlighted if we consider the limitations of simulation in terms of direct optimization. Even when it comes to the use of simulation in off-line manner, and even though time constraints on the decision makers are less tight, time is still an important matter due the fact that the dynamic and competitive nature of modern business imposes on the manufacturing system managers more frequent evaluation of their performance as well as a necessity of maximum control over the manufacturing environments. In other words, modern manufacturers must be able, at any time, to assess their short and long term performance and to react quickly to the rapid, frequent and considerable changes that take place in their environment.

To conclude, we say that simulation offers some interesting possibilities of foreseeing the future at reasonable costs when other exact approaches fail. However, due to its inherent nature (being a trial and error process) and due to the outside constraints imposed by modern business, there is a need to make use of this potential but at a reduced computational requirements. In fact, there is a need for tools that would give some good estimate of the simulation output at reasonable accuracy that would serve at least to reduce the range of decision alternatives (if not to make the decisions directly) and to allow the use of limited number of simulations that would serve as a validation to the estimations made. The work done in this thesis, comes within the framework of making use of the high potential of simulation to capture various aspects of manufacturing systems and of trying to reduce time requirements through the use of neural networks as simulation metamodels. The next section introduces the concept of metamodeling.

2.2 Simulation Metamodels

Simulation has become a widely used and established tool, not only because of its ability to estimate the performance of proposed decisions, but also because of its suitability for sensitivity analysis. Certain analytical techniques, such as linear programming, offer such capabilities at low costs but unfortunately cannot handle all the complexity that exists in modern manufacturing. On the other hand, simulation is able to handle such complexities, but due to its nature, it does not allow itself to perform sensitivity analysis and optimization at low costs. The use of simulation metamodels has been proposed to reduce the computer costs (memory and time) of simulation while making use of its potential of predicting performance of complex systems.

Blanning [3] was among the first to propose the use of metamodels to alleviate the problems related with simulation. The application of metamodels on manufacturing systems is increasing. Yu and Popplewell [35] surveyed 49 papers in this field between 1975 and 1993. Following an early interest in the late 1970, activity fell until 1987. Thereafter, they noted a rapid increase in published work. Yu and Popplewell [35] conclude that the increasing incidence of reported metamodeling in manufacturing-related publications leads to the conclusion that the technique is of value in manufacturing systems design and analysis. However, the review of Yu and Popplewell is based mainly on the regression type metamodels and does not consider the other approaches. Hence, taking the other approaches in consideration, their conclusion is further confirmed.

The simulation model is an abstraction of the real system, in which we consider only a selected subset of inputs. The effect of the excluded inputs is represented in the model in the form of the randomness to which the system is subject to. A metamodel is a further abstraction of the simulation model. It is a model of a model. The selected set of inputs to the metamodel is itself a subset of the inputs considered in the simulation models. Figure 2.1 illustrates this concept. In the abstraction process (i.e. when moving from one level to another), some of the inputs can be either omitted or can be aggregated. Hence,

a metamodel is another approximation of an approximation. It is two steps away from the real system. This means that we cannot expect the metamodel to perform better than the simulation models.

Whenever one is dealing with modeling, the issue of model validity raises. In the case of metamodels, two types of validity should be examined: the first validity is related to the simulation model and the second is related to the real system. According to Blanning [3], the inaccuracy of the metamodel is not very critical and in general will not lead to poor decisions. The inaccuracy will decrease the efficiency of the search for an appropriate decision. The reason for this is that the decision reached by the inaccurate metamodel can be checked by the simulation models and hence an inaccurate metamodel results in increasing the computational efforts caused by the required validation. This reasoning assumes that the validity of the simulation model is guaranteed. This assumption is quite practical, since if the simulation model is not valid then all the analysis will be misleading and there is no need to rely on it. Friedman and Pressman [9] raised the issue related to the validity of the results of the metamodel given the validity of the simulation model on which it was built. Their experiments with regression metamodels have shown that two steps removed from reality, metamodels compared favorably with the true measures of system performance (computed with analytical methods) and with respect to simulation models. Sargent [33] reports some research issues related to regression metamodels which are also valid for other metamodeling approaches such as neural networks. Among the issues raised, are the metamodel validity assessment and experimental design. In our work, we are not concerned with the validity of the metamodel with respect to the real system (as this requires having some real system) but we are concerned only with the validity of the metamodel with respect to the simulation model, assuming that our simulation models are valid models of some hypothetical real systems.

Metamodels have several uses in simulation. It can be used to identify the system parameters that most affect system performance (i.e. factor screening). Since it uses fewer computer resources, the metamodel can be run iteratively many times for repeated *what-if* evaluation for multi-objective systems or

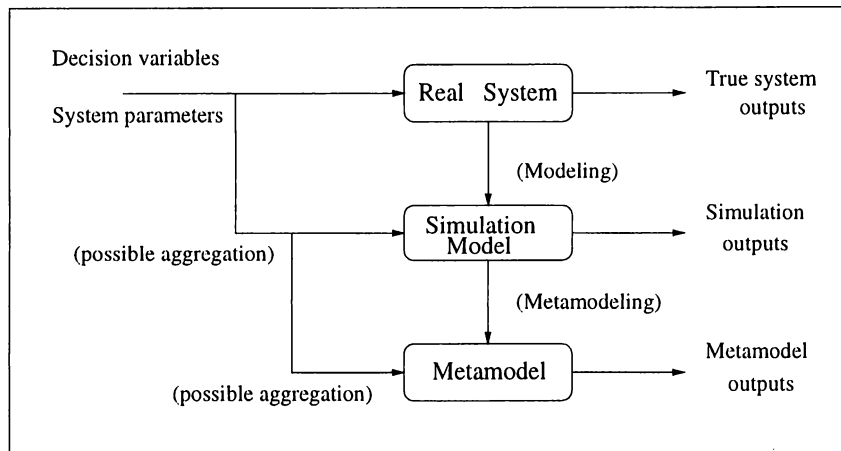


Figure 2.1: Metamodeling Concept.

for design optimization. Another point would be the substitution of the original simulation model by its metamodel when the original model is just one component of a complex decision support system, hence increasing the efficiency of this complex system. This is especially true when the simulation model is incorporated in real time decision support tool where time efficiency is a critical issue. Simulation metamodels provide an approach to summarize the simulation results and allow some extrapolation from the simulated range of system conditions and therefore potentially offering some assistance in optimization. The advantages of metamodeling are explored by Friedman and Pressman [9] based on the regression metamodels. Among these are the model simplification, enhanced exploration and interpretation of the model, generalization to other models of the same type, sensitivity analysis, answering inverse questions and better understanding of the studied system and the inter-relationships of system variables.

Barton [2] reviews the general purpose mathematical approximations to simulation input-output functions. As pointed out by Barton, one of the major issues in the design of the mathematical approximation is the choice of a functional form for the output function. Candidate approaches include: Taguchi models, Generalized linear models, radial basis functions, Kernel methods, spatial correlation models, frequency domain approximations and robust regression methods. Barton concludes that while some approaches are

unable to provide a global fit to smooth response functions of arbitrary shape, the others are computationally intensive and in some cases estimation problems are numerically ill-conditioned. Pierreval has proposed another metamodeling approach based on a rule based expert system [28]. The use of neural networks is another approach for metamodeling which has recently emerged. To our knowledge, a little work has been done to compare the different approaches available and this remains a research direction that has to be investigated. In the work done by Philopoom, Rees and Wiegmann [26], a comparison of regression based due date assignment rules are compared to the use of neural networks for the same task. Their experiments have revealed that neural networks outperformed the regression based rules on two criteria. On the other side, the work reported by Fishwick [8] concludes that neural networks negatively compared with a linear regression model and a Surface Response Model applied on a basic ballistics model (to measure the horizontal distance covered by a projectile). Further investigation regarding the ranking of the metamodeling approaches is required. It is out of the scope of this work to get into the details of these approaches, nor to compare the proposed approach based on neural networks with the previous approaches. This work aims at investigating the approach based on neural networks as it has low computational requirements and does not require some predetermined response function and as it has been reported to provide some good fit in the reported literature.

2.3 Neural Networks

Introduction to Neural Networks

Artificial neural networks take their name from the networks of nerve cells in the brain. The human brain is made of a huge number of simple processing units that individually have weak computing power, but are massively interacting together. This network allows the brain to perform tasks such as image processing and speech recognition that are difficult for

the serial computers. These features allow the brain to accumulate knowledge and respond to stimuli (input) in short times and with relatively high accuracy. Thus, it would be useful to develop an understanding of the mechanisms that govern the functioning of the brain. Artificial neural networks attempt to mimic the parallel and distributed processing that takes place in the brain, although a great deal of the biological details of the brain are eliminated. This simplification is necessary as to allow the analytical tractability of what is happening in the networks.

Dayhoff [7], Masson and Wang [20], and Zahedi [36] provide good introduction materials to the field of neural networks. Basically, an artificial neural network -commonly called neural network- consists of a number of small and simple processing units linked together via weighted and directed connections. Each processing unit receives input signals through weighted incoming connections. The signals are processed by that unit and sent to all the units it has outgoing connections to. Figure 2.2 illustrates a simple example of a three layer back-propagation neural network. Each node in Figure 2.2 corresponds to a processing unit comparable to a nerve cell in the brain. The first layer is the input layer. The second layer is called hidden layer. There can be more than one hidden layer. The third layer is the output layer. The number of units in each layer is a decision parameter. The connections between the processing units are directed arcs. Each of these arcs has an associated weight. Figure 2.3 represents a detailed unit. This figure illustrates the computation that takes place within each unit. Each unit receives inputs x_j 's, along the arcs with weight w_j 's, calculates the weighted sum I of these inputs and applies a transfer function $F(I)$ (activation level). The output of this function, X_i , will be the output of the processing unit. This output is then passed along the arcs connected to this processing unit.

Neural networks are classified based on their learning methods [36][20] into three categories: supervised learning, unsupervised learning and real time learning. Under the *real time learning*, networks continue learning while the network is being used (such as adaptive resonance theory). For *unsupervised learning*, there are no target answers to be achieved by the network. Rather, the

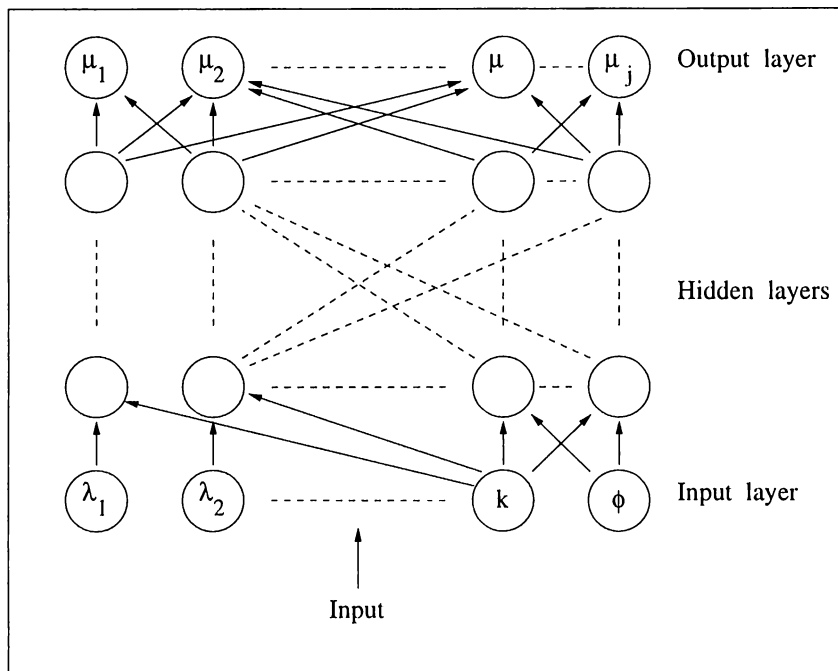


Figure 2.2: The generic network architecture used.

network is trained by learning a pattern through repeated exposure to it and is able to recall the learnt pattern when it solves a categorization or pattern matching problem. For *supervised learning*, a training data set (containing inputs and their corresponding target output) is used to help the network in arriving at the appropriate weights. Back Propagation is the best-known supervised learning method with three or more layers. For this algorithm, input is presented to the network and is propagated forward until it reaches the output layer. At the output layer, the output obtained is compared to the target output corresponding to the given inputs. The error is then propagated backwards along the arcs as to adjust the weights of these arcs. The adjustment takes place according to the Delta rule. The experiments carried out in this work are based on the back propagation algorithm. We consider this algorithm as a black box and we apply it using the NeuralWorks Professional II software [23].

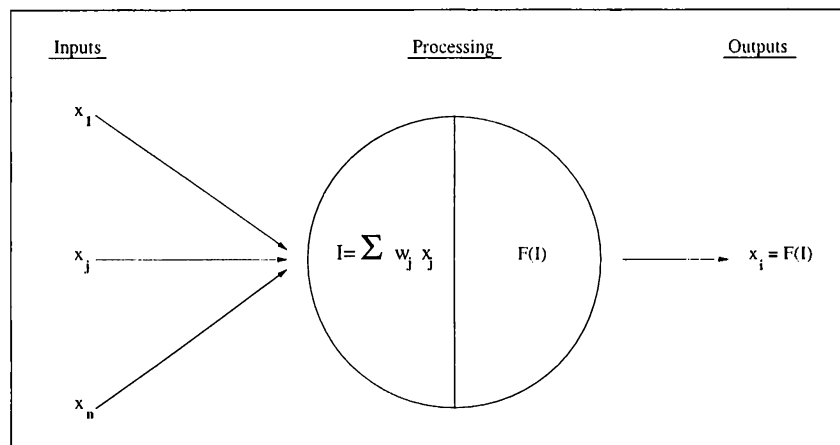


Figure 2.3: Single processing unit in neural network.

Characteristics of Neural Network

Neural networks have been proposed to model systems where the input/output relationship is unknown or too complex; that is to model classes of problems where traditional approaches find it difficult. Therefore, neural networks are not to be used where the already existing approaches perform well since this may result in loss of precision which could be avoided. What distinguishes neural networks from other modeling approaches is their computational speed and learning capabilities as well as their generalization capabilities.

The major distinguishing feature of neural networks is *learning* the underlying mappings between the input and output variables. Traditionally, when modeling systems, the analyst has to provide some input/output relationship and has to test its validity. Neural networks mark a radically different approach to computing compared to traditional methods. For the case of the Back-propagation neural networks, learning is achieved through adjustment of the weights associated with the interconnections of the networks. In a traditional computer program, every step is specified in advance by the programmer. The network, in contrast, would by itself build the mapping describing the input/output relationship and no programming is required. This is achieved by the learning process. Hence, the neural networks can be used to model highly complex systems. In fact, practitioners welcomed Artificial

intelligence (AI), including expert systems, since it allowed consideration of qualitative factors and provided a new approach to incorporate intelligence. Neural networks went a step further with respect to AI. Unlike traditional expert systems where knowledge and intelligence is made explicit in the form of rules, neural networks generate their own rules by learning from examples and extending their knowledge. Although the response function is not explicitly formulated as for analytical metamodels, it is implicitly formulated for the neural network through the architecture applied. These features are likely to give way to including neural networks in expert systems (ES) and thus enhancing the application of ES in manufacturing or in other decision support systems [36][28]. Moreover, neural networks can be tested at any time during training. Hence, it is possible to measure a learning curve of the network. In addition, the network can continue learning even after its actual implementation takes place and the training session has finished. As new input/output examples get available from the real system, they can be presented to the network to improve its accuracy. Also, if some of the system characteristics (that are not given as input to the network) are changing with time (such as improvement in quality), the network can adjust its weights to these changes thanks to its learning capabilities.

Another important feature of neural networks is *generalization*. Although learning is based only on limited set of examples, when it comes to applying the neural network model, the network should be able to extend its knowledge to outside this set of examples. The neural network, if properly trained, can provide correct answers when presented with new inputs that are different from the inputs in the training set. In order to take full advantage of the above mentioned features of neural networks, they must be carefully designed and adjusted to serve the purpose of the study.

Applications of Neural networks in manufacturing environment

Neural networks have a wide range of applications in the manufacturing environments. Zhang and Huang [37] provide a state of the art review of

the applications of neural networks in general. These applications include:

- group technology[21][14][15],
- engineering design,
- monitoring and diagnosis,
- process modeling and control,
- quality assurance,
- scheduling, and process planning.

Burke and Ignozio review the application of neural networks in OR [4]. Udo and Gupta [10] review the applications of neural networks in manufacturing management systems. The applications reported include (in addition to the one mentioned previously):

- resource allocation and constraint satisfaction,
- maintenance and repair,
- database management,
- simulation [30][2], and
- robotics control.

In the survey paper by Udo and Gupta, it appears that the interest in neural networks started mainly since 1987. This corresponds to the same time for which a resurgence of interest was noticed for the use of metamodeling in the manufacturing environments. They also report a list of advantages of neural networks over the conventional computing, such as:

- It has the tolerance to noisy or random inputs.

- It is trained by example and have the ability to adjust dynamically to changes in the environments.
- It has the ability to generalize from specific examples.
- It has a slow degradation in problems outside the range of the experience.
- It has the ability to discover complex relationships among inputs variables, and
- It has speed of response.

Constructing Neural Networks for Simulation Metamodeling

Many design issues are involved in developing a neural network metamodel. Care must be given to these issues as they are essential in developing a reliable and robust neural network metamodel. This is especially important as the metamodels are models of simulations model [3][33][35][24]. Hence, the error of the network with respect to the real system will be amplified if the network is not properly designed. Whether it is appropriate to use a metamodel or not, is a matter that depends on the application and how much approximation is acceptable. However, it appears that increasingly more people are making use of metamodels [9]. Figure 2.1 illustrates two main issues involved. In addition to selecting the appropriate variables for the application under consideration and to constructing a valid simulation model, the metamodel itself is a major issue. We have to decide on the internal parameters of the metamodel. Khaw, Lim and Lim [17] report an optimal design of neural network models based on the Taguchi method in terms of setting the internal parameters of the model for the back propagation-type networks. They claim that their approach improves network reliability and convergence speed. Other authors have selected other approaches. For our experiments, we did not put much emphasis on this part as our aim was not building very precise networks but rather examining their behavior.

As mentioned previously, the way the metamodel is constructed has a

significant impact on its performance. The following is a general design procedure for metamodeling with neural networks. As can be seen, this procedure does not differ in much from other metamodeling approaches.

- Step 1: Define the system: inputs, outputs, parameters, performance measures and mechanisms governing the relationship between inputs and outputs.
- Step 2: Develop a valid simulation model to examine the performance of the system under some experimental conditions.
- Step 3: Select the set of variables that will be considered by the network as inputs. These usually include the decision variables and system parameters that are expected to be varying during the period of study. Decide on how the performance (or the validity) of the metamodel will be evaluated.
- Step 4: Decide on how these inputs are to be presented to the network since the input data may need some preprocessing [17].
- Step 5: Decide on the internal design of the neural network. This includes deciding on the number of layers, the number of processing units per layer and the interconnections (full connection, partial connections), etc. [17][29].
- Step 6: Select the network paradigm that would control the processing that takes place in the processing units and the training procedure. There are a number of paradigms available such as back-propagation (widely used in manufacturing applications). Each of these paradigms has several parameters that need to be fine tuned to ensure the appropriate learning and performance of the network.
- Step 7: Once the above issues have been decided upon, training can start. Develop a training set using the simulation models and perform the training. Several iterations may be required between steps 5, 6 and 7 in order to find the best neural network with the least errors.

- Step 8: Validate the designed neural network using a test set that contains examples not included in the training set.

Training can continue even after the network has been validated. As new examples from the real system become available, the network can be trained on them; thus further reducing the error with respect to the real system.

In evaluating the precision of the built metamodel, several candidate error measurement methods can be available. It is essential to select an appropriate one. As our experiments have shown, the constructed metamodels may have a different ranking based on the evaluation criterion applied. Therefore, we recommend that a great care should be given to this issue. The importance of this issue is discussed in detail later in the text.

The most widely used implementations of neural networks are software simulators. These simulate the operations of the network on serial computers as these are very much available at low prices. However, the time requirements for developing and implementing the neural networks could be further reduced if hardware with parallel processors are used. Thus, the full potential of neural networks can be further enhanced with developments in hardware.

Drawbacks of Neural Networks

Several shortcomings related to the current applications of neural networks as a metamodeling technique have been reported in the literature [19]. First, constructing a neural network is time consuming as this process requires generating a training set, empirically selecting an appropriate architecture and learning algorithms. Secondly, the accuracy of the network outputs depends on the regularity of the behavior of the system under study (by regularity we mean that the system is subject to the same set of exogenous and uncontrollable factors). This implies that the time horizon of the study must be carefully selected. Thirdly, the validity of the results depends also on the degree of aggregation selected for the input data. Aggregation of data is needed in order

to reduce the size of the neural network and the effort required to generate the examples. This would have a negative impact on the precision of the neural network results. The disadvantages mentioned so far are common to most metamodeling techniques.

Another more specific problem related to metamodeling with neural networks is the difficulty to make interpretations and analysis of the input/output relationship. As mentioned previously, the neural network generates its own rules but does not provide them explicitly to the user. In order to get an insight into the input/output relationship, one needs to analyze the weights of the connections between the processing unit. This is not an easy task, and it is time consuming. Thus, providing a formal method to analyze the neural network may strengthen its value as a metamodeling approach. Furthermore, the selection procedure for the network architecture, learning algorithm and parameters is in most of the reported cases a trial (empirical) process. Some attempts have been made to provide a formal approach to do this task. Khaw, Lim and Lim [17] propose a method based on a Taguchi approach. Murray [22] used genetic algorithms to perform this task. Further deficiencies in the literature are concerned with the lack of development of learning algorithms. Research in this direction may allow more exploration of the full potential of neural networks.

2.4 Neural networks as a simulation metamodeling approach

Our research is focused on simulation metamodeling with neural networks for the purpose of estimating system performance measures. Zhang and Huang [37] have reported an increasing interest of the use of neural networks in the manufacturing environment since 1987. Starting the same period, Yu and Poplewell [35] report an increasing interest in simulation metamodeling. This illustrates that these two different techniques have an increasing potential of contribution to improving the management of manufacturing systems. Despite

this interest, efforts to combine metamodeling and neural networks through the use of neural networks as a simulation metamodeling approach has not been much. In fact, for this type of applications of neural networks, the related literature is not abundant. Seven papers applying the back propagation neural networks as a simulation metamodel for the management of manufacturing systems are surveyed.

Chryssolouris et al. [6] used a neural network metamodel to reduce the computational efforts required in the long trial process that is associated with using simulation alone for the design of a manufacturing system. The simulation model is used to generate the performances (4 performances measures are recorded) of the system under different designs. The neural networks is then used in an inverse manner. The input of the neural network is the desired levels of the performances of the system and the output would be the design that would achieve those levels of performance. Although this application was successful, some questions were raised regarding the complexity of the system and regarding the complexity of the application itself. In fact, because of the small size of the system considered, the number of design alternatives is not large. However this application, indicates a potential use of neural networks as system design has an important impact on its performance and often the design phase is time consuming because of the large number of alternatives.

Simulation metamodeling with neural networks mostly is applied as a tool for determining operational policies since it is in this type of applications that time is more crucial. Chryssolouris has developed a task assignment procedure that is based on multi-criteria, called MADEMA (MANufacturing DEcision MAKing). This approach combines the system performance criteria according to some given weights. Chryssolouris et al. [5] used a neural network metamodel to determine the weights required to achieve some given levels of the multiple criteria. Although the application showed some good results and a good ability of neural networks to handle complex relations, one may question the effect of the small range of the inputs and the effect of system complexity. Hurrion [13] used a neural network to estimate confidence intervals

for the performance of an inventory depot. This application revealed that neural networks were equally successful to estimate mean performance as well as their corresponding confidence intervals. Moreover, this work highlighted the capabilities of neural networks to model problems with large range of inputs and complex input/output relation but still does not provide an insight on the effect of system complexity nor on the effect of stochasticity. Another case was examined by Pierreval [27] to investigate the ability of neural networks to estimate mean machine utilization of a deterministic small sized problem. The results were encouraging as in this problem input range was wide and also the neural networks showed its ability to learn and generalize properly. The questions that raises here may be regarding the effect the performance measure, stochasticity and system complexity. Pierreval [29] later proposed a neural network architecture to be used for ranking the performance of dispatching rules on a stochastic flow shop type system. Neural networks have performed well and highlight the modeling flexibility that modeling with neural networks can offer. Here one may question the effect of system configuration (flow shop vs. job shop). The work reported by Philipoom et al. [26] gives some other type of application of neural networks as a simulation metamodel. Neural networks are applied to assign due dates for jobs based on system characteristics and system status when jobs enter system. The use of neural network for individual jobs contrasts with the use of neural networks to get aggregate system measures (such as mean flow time in the previously mentioned publications). The performance of neural metamodels compared favorably with regression based metamodels and showed another interesting type of application. Kilmer et al. [18] report a possible use of neural metamodel for a service activity, an emergency department. They tested the validity of metamodel, with respect to the real system, and it appears that the validity is high.

In the seven publications reported in the last 2 paragraphs, the constructed neural network metamodels achieved reasonably good results. The authors showed that neural networks are a very promising tool for predicting system measures. However, these case studies deal with systems of reduced complexities or of deterministic nature and do not allow us to generalize on

the estimating capabilities of neural networks. The following set of questions summarizes some of the future research issues that still need to be investigated:

- How to assess neural metamodel performance?
- What is the effect of system size on the performance of the neural network?
- To what extent can the neural network handle system stochasticity?
- Do stochastic factors effect differently network performance?
- Is the metamodel performance affected by the fact that the system is in transient state or in steady state?
- Is the performance of the metamodel affected by the level of activity of the system?
- What is the effect of the performance measure being predicted?
- What is the effect of the network configuration (size, number of layers, learning rate ...) on the performance of the network?
- Does system configuration (flow shop-job shop) have an effect on the performance the developed neural networks?
- How robust is the neural metamodel to noisy data, to data outside the training range?
- How adaptive is it to gradual small changes in the system over time (such as gradual improvements in quality)?
- What are the computational requirements in terms of computer time?
- How to select the size of training and test data?
- How to choose simulation run length?

This small set of questions is representative of the current vacancies in the literature. In this work, we don't attempt to answer all these questions. Rather we concentrate only on the first eight questions. We don't intend to give extensive and final answers to these questions. We aim at constructing experiments that would allow us to get an insight on these issues. As a matter of fact, our work investigates two types of application of neural network metamodels: estimating long term system performance and short term system performance.

For the first application, we will investigate the effect of:

- * Performance measure: Mean machine utilization and mean job tardiness.
- * System complexity: Simple vs. complex system.
- * Stochasticity: deterministic, stochastic interarrival times only, stochastic processing times only or both stochastic.
- * Demand on system: low, medium and high.
- * Metamodel error assessment criteria.

In all the reports we described previously, neural metamodels are examined in terms of estimating long term performance. and the available literature is not abundant yet. For estimating short term performance, we could not find any reports on the use of neural metamodels for such applications. For this we also examine such an application in order to gain insight on the effect of this issue and so we can get an idea about the possible use of neural metamodels for real time decision support. With the second application, we will consider the mean job tardiness as a system performance measure and we allow the system to be deterministic or to be subject to stochastic processing times and interarrival times. That is, we investigate the effect of:

- * Initial system status.
- * Demand on system: low, medium and high.

* Metamodel error assessment criteria.

For both applications, we preview the effect of due date tightness factor and of the effect of the size of training set. In order to preserve a basis of comparison, we use the same system structure, neural network architecture and same error assessment criteria. The neural network learning algorithm is considered as a black box and we don't attempt to improve it. We also do not carry out extensive fine tuning of the parameters because it is time demanding and because our emphasis is more on developing an understanding of the behavior of neural metamodels with respect to the factors previously mentioned. The next 2 chapters investigate long term mean machine utilization and mean job tardiness respectively. The fifth chapter examines the situation for the short term mean job tardiness. Finally, we give our conclusions and future research directions in the last chapters.

Chapter 3

Estimating Long Term Machine Utilization

In this chapter, we investigate the capabilities of neural metamodel in estimating mean machine utilization as a system performance measure. We consider two job shop systems, which we would refer to as case one and case two. The first job shop system is a simple one with four machines and three distinct product types. The second system is a complex one. This system can be considered as an extension of the first system, to include more machines and more product types. This extension is made in such a way as to keep a basis of comparison between the two cases. This is achieved by adding three more machines and three more job types to the first system. While the job types common to both systems keep the same parameters in terms of processing and routing requirements, the new product types have processing requirement on both the old and the new machines. Hence, the first system is a subset of the second one. This allows us to investigate the effect of increased system complexity by studying the second system and comparing it to the first one. Therefore, the term “system complexity” in this study would refer to increased system size (increased number of machines and increased number of job types) as well as increased interactions between the different components of the system. For each of the two cases, we describe the experimental settings

through describing the system, the simulation models, the neural network metamodels and the error assessment approaches. The next section lays down the results and discussions. The last section of the chapter compares the two cases.

3.1 Case 1: Simple System

In this first case, we consider the work reported by Pierreval [27] as the starting point. His work tries to estimate mean machine utilization for a deterministic system. In the first step, we simply repeat this work. In our experiments, however, the back propagation learning algorithm is improved by adding a momentum term. The second step is to investigate the stochastic configurations of the same system (stochastic arrival times only, stochastic processing times only or both) and to test the robustness of the metamodels designed for the deterministic configuration to inputs that lie outside the training range.

3.1.1 Experimental settings

System description

This study is based on the work done by Pierreval [27]. His experiment consists of running a simulation model for a deterministic job shop system. Based on this model, a neural network metamodel to estimate machine utilization is constructed. This job shop system, he used, is deterministic. It is composed of four machines and three free transporters. Three job types are entering the system. Jobs arrive independently to the system at constant rates: λ_1 , λ_2 and λ_3 . Jobs await for the availability of machines in queues according to a waiting discipline ϕ . The waiting discipline, ϕ , could be either Shortest Processing Time (SPT) or First Come First Served (FCFS).

In this first case, we consider four possible configurations of this job shop system: a deterministic configuration, a configuration with stochastic interarrival times only, a configuration with stochastic processing time only and finally a configuration with stochastic interarrival times and processing times. Our work is based on the same system. All the relevant data can be obtained from the sample codes provided in Appendix C. For the case of stochastic arrival times, those same values of the constant interarrival times λ_1 , λ_2 and λ_3 are used as the means of the corresponding exponential distributions. Similarly, the constant values of the processing times, used in the deterministic configuration, are used as means of the corresponding exponential distributions in the stochastic processing times configuration. The choice of the exponential distribution for both factors (arrival and processing times) appears reasonable since a large number of the reported simulation experiments use this distribution which seems to match real life as well [1] [32].

Simulation Models

The simulation models of the system described above are developed and used to run the job shop system for various configurations. The production is performed in two shifts. For the deterministic configuration, the model is run during one week of work (5 days), plus one day of transient phase. For the stochastic case, a transient period of 2 work days is used, and the system is run for 15 days, to form 5 batches of three days each (Batch means approach is used through out this study). We are interested in finding the average machine utilizations μ_1 , μ_2 , μ_3 and μ_4 of the four machines in order to detect bottleneck machines as well as under-utilized machines. Therefore, given an input combination of λ_1 , λ_2 , λ_3 and ϕ , we run the corresponding simulation model to record the output combination μ_1 , μ_2 , μ_3 and μ_4 ($\mu_i \in [0, 1]$). These outputs are the true values of these variables that the neural metamodel has to estimate. The combinations of these inputs and outputs would compose one example in the data set that is presented to the neural networks either in a training set or as a test set. The simulation models are developed in SIMAN language [25]. Sample model and experimental frames are provided

in Appendix C. Four simulation models are used; each corresponding to one of the configurations mentioned above. Table 3.1.1 in Appendix B shows the list of the models built. Figure 3.1.1 illustrates the relationship between those models. For each model the corresponding training and test sets are generated. In terms of machine utilization, the characteristics of the data sets generated from these models are quite identical for all the models examined in the first case, and are as follows:

- minimum utilization: 14.5%
- mean utilization: 45.2%
- standard deviation of utilization: 17.7%
- maximum utilization: 100%

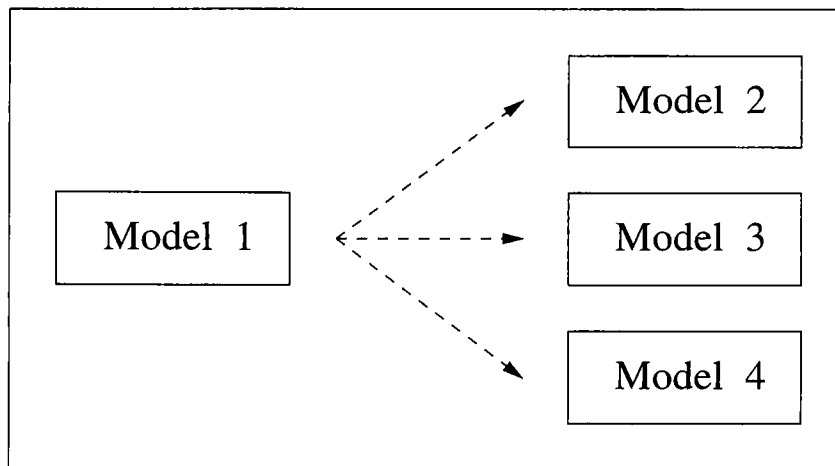


Figure 3.1: Mean Utilization: Simple System: Relationship between models.

This means that independently of any factor being stochastic or deterministic, the resulting sets are similar. This may lead us to think that similar results can be obtained from all the models.

For model 1, we also develop test Set #5 which consists of 50 examples and is similar to test Set #1, where interarrival times are deterministic, but this time are randomly selected from $[85, 100]$ instead of $[10, 85]$. We also

generate Set #6 which consists of 50 examples and is similar to test Set #1, where interarrival times are deterministic, but this time are randomly selected from $[100, 120]$. Test sets #5 and #6 are used to test the robustness of the metamodel with respect to inputs that lie outside the range of the training data and hence allow evaluating generalization capabilities of neural metamodels. Two sets are used in order to see how the performance of the neural metamodel evolves as we move far from the range of the inputs of the training set.

As each data set (training or test sets) require a set of examples, we need to create the set of inputs to be given to the simulation model in order to generate the true values of the variables of interest (mean machine utilizations), and to be presented to the neural network in order to generate estimates of the variables of interest. A SIMAN code was used to randomly generate the values of the inputs in the desired range from a uniform distribution. Another approach would have been to generate these inputs using experimental design techniques. However, the first approach was used because it corresponds to real life more where examples would follow a random scheme. Appendix C shows the model and experimental frames for this input data generator model.

Neural Network Metamodels

Several Back-propagation neural networks are designed with various architectures (number of processing units and layers) and various combinations of network parameters (learning rate, momentum term). No bias is introduced, nor dynamic adjustment of the learning parameters are used. The sigmoid function is used as the transfer function in the processing units. Inputs are scaled in the interval $[0, 1]$. Table 3.1.2 in Appendix B shows the characteristics of the networks constructed for each model. In this Table, the name assigned to each neural network are as follows: Exp1_A.B. This coding should be read as the name of the neural network number B developed as a metamodel for model A of this first set of experiments. An example would be: Exp1_2_3. This describes the network number 3 designed for model 2 of this first set of experiments. Refer to the generic architecture given in Figure 2.1.

The following generic architecture is used:

- Input layer: 4 processing units:
 - Interarrival time for job type 1, λ_1 .
 - Interarrival time for job type 2, λ_2 .
 - Interarrival time for job type 3, λ_3 .
 - Dispatching rule, ϕ : 1 or 2 (SPT,FCFS).

(for the stochastic case, the mean of the distribution is presented as input)

- Hidden layers: Various combinations of networks with different number of hidden layers, with different number of processing units are used.
- Output layer : 4 processing units:
 - Machine 1 average utilization: μ_1 .
 - Machine 2 average utilization: μ_2
 - Machine 3 average utilization: μ_3 .
 - Machine 3 average utilization: μ_4 .

The neural network model was developed using a network simulator NeuralWorks Professional II developed by NeuralWare, Inc [23]. It is necessary to mention also that further fine tuning of the parameters of the neural networks built, may lead to better results but it is believed that this improvement would not alter the conclusions obtained.

Error Assessment

One essential aspect of metamodeling is how to evaluate the error of the neural network metamodel with respect to the desired performance. For this purpose,

four different evaluation methods are used. Which assessment approach to use when it comes to real life application of neural metamodels, would depend on the objectives of such an application. However, by the use of multiple criteria, we intend to investigate the robustness of the neural metamodel performance to multiple criteria. Moreover, consistency through out the criteria would add to the reliability of the conclusions made.

Let

i : be the index to represent the example number in the training or test sets.

j : be the index to describe machine number. $j = 1, 2, 3, 4$.

S_{ij} : be the average utilization obtained for machine j at example i obtained from the simulation model (true value of the variables of interest).

N_{ij} : be the average utilization obtained for machine j at example i obtained from the neural network metamodel (estimate of S_{ij}).

T : be the total number of examples in the training or test set.

α : Tolerance level, $\alpha \in [0..1]$.

We also define:

$D_{ij} = |S_{ij} - N_{ij}|$: absolute deviation for machine j and per example i .

$D_i = \max\{D_{ij}, j = 1, 2, 3, 4\}$: maximum absolute deviation across all machine for example i .

- **Method 1: (Tolerance approach)**

This method has been used by Pierreval in [27]. The performance of the neural networks is evaluated according to the percentage of examples not recognized . An example is said to be recognized if the maximum

absolute deviation among the four output variables is less than a given tolerance level, α , that is:

let:

$$I_i(\alpha) = \begin{cases} 1 & \text{if } D_i > \alpha \\ 0 & \text{otherwise.} \end{cases}$$

From the data set under consideration, we calculate the average error, $E(\alpha)$, where:

$$E(\alpha) = \frac{\sum_{i=1}^T I_i(\alpha)}{T}$$

α is allowed to take values in $\{5\%, 6.5\%, 8\%\}$.

- **Method 2: (MAD approach)**

This method measures the mean absolute deviation, MAD, across all the examples in the data set under consideration and across all the variables being estimated (machine utilization in this case) and the corresponding standard deviation, where:

$$MAD = \frac{\sum_{j=1}^4 \sum_{i=1}^T D_{ij}}{T}$$

- **Method 3: (MMAD approach)**

Another method is to measure the error of the network in terms of the mean maximum absolute deviation, MMAD, across all the examples in the data set under consideration and across all the variables being estimated (machine utilization in this case) and the corresponding standard deviation, where:

$$MMAD = \frac{\sum_{i=1}^T D_i}{T}$$

This method is similar to the second method, except that it is more severe and penalizes the neural metamodel for the highest deviation through each example in the data set.

- **Method 4: (Percentage error approach)**

This method is the classical one. It is based on evaluating the relative error for each example and for each variable in the data set and taking the average, E_j , and the corresponding standard deviation.

$$E_j = \frac{\sum_{i=1}^T \left(\frac{D_{ij}}{S_{ij}} \right)}{T}$$

Evaluating the metamodels based on the first three measures is a subjective matter since we are dealing with absolute measures. The interpretation of these methods requires a prior knowledge about the system parameters. The fourth method is used to assess the real life applicability and the acceptability of the results, where as the three first are used to highlight the differences between the models constructed. The fourth approach is a relative approach and hence it is objective. We try to use the combination of those methods to come out with satisfactory metamodels.

3.1.2 Results and Discussions

Table 3.1.3 in Appendix B presents the results achieved by each neural network metamodel for each of the models investigated across the four error assessment methods discussed above. For each model, we select the best metamodels among the set that was built (shown in gray background). Given these “best” neural metamodels, we graph their performances relative to the four evaluation criteria previously defined. These graphs are provided in Figure 3.1.2 in Appendix A, both for the training sets and the test sets. The term “best” network refers to the one that outperforms the others on the majority of the error assessment methods, with the highest weight given to the fourth criteria (the percentage error approach).

Model 1 (Deterministic interarrival and processing times) is a replicate of the work reported by Pierreval[27] except that we include a momentum term

in the back-propagation algorithm. The results achieved by Pierreval are given in Table 3.1.4 in Appendix B (based on the first evaluation method). The results corresponding to this case are given in Table 3.1.3 in Appendix B for network `exp1_1_2`. We can see that there is an improvement in those results which can be explained by the improvement in the learning algorithm. For the purpose of illustration, we include Figure 3.1.3 in Appendix A which shows the learning curve of this neural metamodel (`exp1_1_2`) based on the first evaluation method.

We also test the ability of the neural network used for Model 1 (`exp1_1_2`) to extend its fitting of the response function (mean machine utilization) to outside the range of the input values used while training. Table 3.1.5 in Appendix B, shows the results of two of experiments (robustness experiments). In the first experiment, the input data are generated from an interval outside the range of training but not far from it. In the second experiment, input data are generated from an interval far from the range of study. Here, we evaluate the performance of the network based on the first evaluation method. In both experiments, when considering a tolerance level, α , set at 10% the network error is 0%. However, for α at 5% the error sharply increases. That is to say that in most of the tested examples, the error was confined to some small interval [5%, 10%]. Further investigation regarding metamodeling robustness, as defined above, is required. Such an investigation of the data sets may lead to define the appropriate size of the range of the inputs without major loss of accuracy.

For the stochastic configurations (Models 2, 3 and 4), it appears that the precision of the neural metamodels decreases as we introduce sources of stochasticity, as can readily be seen from the graphs in Figure 3.1.2 in Appendix A. This behavior can be expected since including stochastic factors in the system means introducing sources of irregularity in the patterns of behavior of the system. Since neural networks operate by trying to classify patterns of behavior, irregularities cause some difficulties for the network to classify inputs.

Another observation which was not expected, is that, independently of

the error assessment criteria being used, the network precision does not vary significantly if we move from Model 2 to Model 3 (see flat middle portion on the graphs). Thus, introducing stochasticity on the interarrival time factor or on the processing time factor, would produce a comparable effect on the patterns of behavior of the system and hence a similar effect on the accuracy of the metamodel. Therefore, it seems that the number of stochastic factors is more important than the nature of the factors themselves. Despite the fact that stochasticity in the system causes the accuracy of the metamodel to estimate mean machine utilization to deteriorate, this accuracy remains within acceptable limits. We use multiple evaluation criteria and we find that these conclusions are valid across all those criteria.

For error assessment criteria 2, 3 and 4, the experiments have shown that in moving to more complex system configurations (through the addition of stochastic factors), the performance of the metamodel does not deteriorate further as we move from the training set to the test set. This implies that generalization capabilities are not affected by stochasticity. However, for the first evaluation criteria (the tolerance approach), this is not the case for the tight tolerance level case ($\alpha = 5\%$). The reason for this is that we selected the values of the tolerance levels, α , based on the training sets in such a way as to highlight the effect of this tolerance level. Thus, as we move to the generalization on the test set, the slightest deterioration will be revealed mostly by the tight tolerance level ($\alpha = 5\%$). As can be observed, the generalization capability is not affected when it comes to the other levels of tolerance ($\alpha = 6.5\%$ and 8%). This indicates that the deterioration in metamodel precision, as we move to the test set, is limited and can be considered as acceptable.

The experiments have also shown that metamodel performance is consistent through out the four criteria. That is to say that metamodels have the same ranking on the four criteria. On the other hand, no architecture is best for all models. Thus, in order to find the best metamodel, unfortunately, a large number of networks have to be constructed and tested. This would result in a long metamodel development phase.

There are a few observations that are worth mentioning regarding our experiments. First, from the different neural networks developed, it appears that the way in which the input is presented to the network (random or sequential) did not have a considerable effect on the precision of the results, although sequential access seemed to produce slightly better results. Second, from the various experiments carried out, it appears that the initialization range of the weights on the arcs of the network did not effect the precision of the results, if we allow sufficiently enough long training period. Third, the relationship between the learning rate (and also the momentum rate) and the network precision is not linear. Fourth, in some cases the network performed better on the test set than on the training set itself. Although this may be unexpected, it may be due to the randomness of the test sets. Experimenting with neural networks has indicated that they can handle noisy training inputs. In fact, training the network with some data set which contained a portion of wrong examples, resulted in a network able to achieve 2 to 3% error on Method 1.

3.2 Case 2: Complex System

The system investigated in this case is similar to the system examined in the previous case, except for the complexity introduced as explained at the beginning of this chapter. The way in which we increased system size (in terms of more job types and more machines) ensures us that this second system is more complex and hence tends more to real life systems. Because of this increased complexity, the study will be more involved as the performance of the neural metamodel is evaluated at different levels of demand on system. The term levels of demand on system refers to the ranges from which interarrival times are generated.

3.2.1 Experimental settings

System description

In fact, the system considered in the previous case is a subset of the system considered here. The job types and the machines simulated in the first case are also present here with the same parameters. That is to say, the old job types keep the same processing times and routings and the old machines keep the same capacity and are still located at the same distances from each other. Three new job types are added, to form a total of six, and three new machines were included. The new job types would have a routing on both old and new machines. However, in order to keep the system balanced, these new job types have smaller processing times on the old machines than on the new machines. This system is allowed to operate under three job waiting disciplines, ϕ : SPT, EDD and Modified operation Due Date (MOD). This increase in the number of possible dispatching rules that can be used in the system is another dimension of complexity that's added to this study. The method used to assign due dates is the Total Work content method (TWK) and it involves a due date tightness factor, k , which is generated from the range [2, 9]. This range covers the tight and loose due date situations. The TWK has been supported as among the best due date assignment rule in job shop environments [1] [32]. Like the previous case, this system also operates with three free transporters. Each job type arrives to the system with a constant interarrival time λ_i ($i = 1, 2, \dots, 6$) for the deterministic arrival configurations. For the stochastic interarrival cases, the same values of λ_i are used as the means of an exponential distribution. Similarly the values of processing times used in the deterministic models, are used in the stochastic models as the means of the corresponding exponential distributions (as in the previous simple system case).

Simulation Models

Simulation models of the system described above are run under various configurations (deterministic and stochastic) and under different values of interarrival times and under different dispatching rules. Production is performed in two shifts. For the deterministic models, the system is run for 2 days as of a transient period and statistics are collected for 8 days. For the stochastic models, the batch means approach is used with 5 batches. The transient period has a length of 4 days and each of the 5 batches has a length of 3 days. Note that the lengths of the transient periods, in this complex system case, are doubled relative to the first case (simple system) in order to allow the system to reach its steady state. Here also, for a given input combination of λ_i, k and ϕ , we run the simulation models to record the mean machine utilization, μ_j , of the 7 machines. Input data (λ_i, k and ϕ) are generated with a similar input data generator to the one used in the first case. Table 3.2.1 in Appendix B shows the list of the models built. Table 3.2 in Appendix B shows the characteristics of the data sets generated from each model in terms of their machine utilization and in terms of the fraction of tardy jobs. Figure 3.2.1 illustrates the relationship between these models.

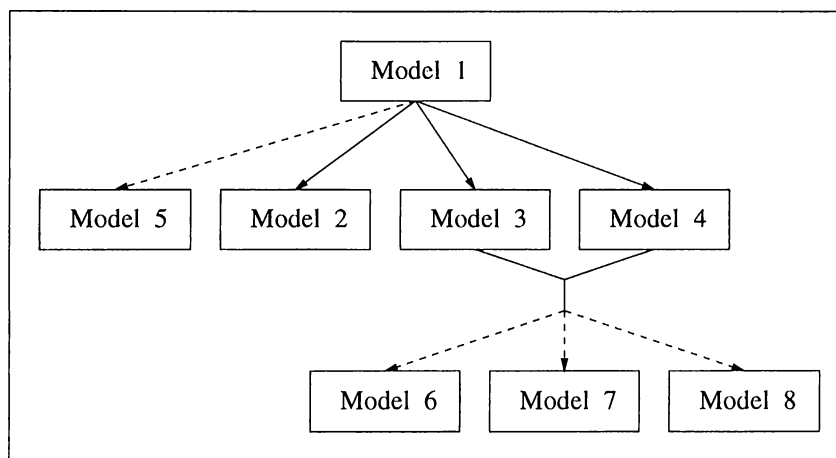


Figure 3.2: Mean Utilization and Mean Tardiness: Complex System: Relationship between models.

Neural Network Metamodels

For each model, we run three neural network metamodels. Only the network architecture (number of hidden layers and number of processing units per layer) is changed. The network learning rate and momentum term are not changed as it would require a very long time to cover all the space of these parameters. These neural networks are very similar to the ones detailed in Section 3.1.1. Table 3.2.3 in Appendix B shows the list of the neural networks built for each model.

The following generic architecture is used:

- Input layer: 8 processing units:
 - Interarrival time for job type 1, λ_1 .
 - Interarrival time for job type 2, λ_2 .
 - Interarrival time for job type 3, λ_3 .
 - Interarrival time for job type 4, λ_4 .
 - Interarrival time for job type 5, λ_5 .
 - Interarrival time for job type 6, λ_6 .
 - Dispatching rule, ϕ .
 - Due date tightness factor, k .

(for the stochastic case, the mean of the distribution is presented as input)

- Hidden layers: three combinations of networks with different number of hidden layers, with different number of processing units are used.
- Output layer : 7 processing units:
 - Machine 1 average utilization: μ_1 .
 - Machine 2 average utilization: μ_2

- Machine 3 average utilization: μ_3 .
- Machine 4 average utilization: μ_4 .
- Machine 5 average utilization: μ_5 .
- Machine 6 average utilization: μ_6 .
- Machine 7 average utilization: μ_7 .

Error Assessment

In order to evaluate the performance of the neural metamodels, the same error assessment approaches applied in the simple case is used (refer to Section 3.1.1).

3.2.2 Results and Discussions

The results of this second case are given in Table 3.2.4 in Appendix B. This Table shows the performance of all the neural metamodels relative to the four error assessment methods selected. Here also, the performance of the best neural network metamodel corresponding to each of the 8 models investigated, are graphed in Figure 3.2.2 in Appendix A. The term “best” network refers to the one that performs better than the others on the majority of the error assessment methods, with the highest weight given to the fourth criteria (the percentage error approach). The fourth criteria is the only objective criteria and does not require any prior knowledge of the characteristics of the system and hence allows direct and objective interpretation of the performance of the metamodels.

Recall that in the simple system case we had only investigated one deterministic model (Model 1). Here, Models 1 through 5 are deterministic models. For these five models, the neural networks performed well on all the error assessment approaches. Models 2, 3 and 4 are subset of the first one. The aim is to view the effect of reducing the size of the range from which interarrival times are generated as well as the effect of the *demand on system*. The term

demand on system refers to the frequency with which jobs arrive to the system, which is determined by the interarrival times. It appears that the predictive capability depends on the ranges themselves. As we decrease the demand on system, that is going from Model 2, to Model 3 and to Model 4, network precision increases. This means that neural networks perform better for systems with low demand than with higher demand. This could be explained by the fact that increased demand adds more interactions in the systems, and hence makes it more difficult to classify system behavior into patterns. Since neural networks operate via pattern classification, network precision deteriorates, at least slightly when the demand on the system increases under consideration.

Model 5 is identical to Model 1, except that all the examples in the data sets, for which the average machine utilization is above 98%, are removed. For such examples, the system may be out of its steady state. Hence, this experiment would show if including examples which correspond to the system being out of its steady state in the data set, has an effect on the neural network performance. A slight improvement has been noticed for Model 5 over Model 1. In fact, the results achieved by Model 5 are comparable to the ones achieved in Model 2. Although the data for these two models is generated from different ranges (interarrival times for Model 5 are generated from [20..100] and for Model 2 from [20..40]), the models performed comparably and so we can say the length of the ranges did not affect the network performances, but rather it is those examples that correspond to extremely loaded system that deteriorate most the metamodel accuracy.

For the stochastic configurations (Models 6, 7 and 8), it appears that the precision of the neural metamodels significantly decreases as we introduce sources of stochasticity, as can readily be seen from the graphs in Figure 3.2.2 in Appendix A. Moreover, it appears that network precision is more affected by the processing time as a stochastic factor than by the interarrival time as a stochastic factor. However, the difference between these two factors is not very large if we compare it to when both factors are stochastic. Although the three stochastic configurations perform significantly worse than the deterministic

cases, they still can be considered to be acceptable for all the error assessment criteria except for the tolerance approach (Method 1). Nevertheless, this tolerance approach imposes very tight tolerance levels and so we cannot rely on it to judge on the performance of the networks, because if we slightly relax the tolerance levels, the performance will significantly improve.

Concerning the generalization capability of the neural networks, they appear to be acceptable, except for the first error assessment approach because of the tight tolerance levels imposed. The experiments have shown that metamodel performance was consistent through out the four criteria. In other words, metamodels have the same ranking on the four criteria. On the other hand, no architecture is best for all models. Thus, in order to find the best metamodel, unfortunately, a large number of networks have to be constructed and tested. This would result in long metamodel development phase.

3.3 Comparison of Simple vs. Complex systems

Although in both cases, the neural network metamodels proved to perform well in estimating mean machine utilization, there are several remarks worth mentioning regarding the effect of system complexity. First, the increased complexity acts more negatively on metamodel performance as we move from the deterministic to the stochastic configurations. The first graph on Figure 3.3 highlights this fact. This could be explained by the fact that stochasticity is itself another dimension of complexity, and if it is combined with system complexity then the system would undergo more interactions that are difficult to classify. The second point concerns the stochastic factors. It appears that introducing system complexity leads to the factor of stochastic processing times to have a slightly more negative effect on neural network performance than the factor of stochastic interarrival times. Concerning the generalization capabilities, they slightly decrease with increased system complexity. However, the magnitude of this deterioration is nearly constant as we move to stochastic

configurations. This can be seen from the second graph on Figure 3.3. Also, demand on system appears to be an important factor to metamodel accuracy, the higher the demand the lower the accuracy.

The error assessment methods produced consistent results in both cases. In other words, the different models would rank similarly on all the methods. This consistency is a good point as it confirms the conclusions made so far. On the other hand, if we consider the tolerance approach (Method 1), the performance of neural network metamodel for the stochastic configuration may not be considered as acceptable. This can be explained by the tight tolerance levels applied. However, this requires some subjective judgments that depend on the applications of this type of metamodeling and the objectives of the application.

To conclude, we can say that complexity, whether it is due to stochasticity or to the system, results in deteriorating neural network learning and generalization capabilities. Despite this, the results achieved are good enough to show that neural metamodels are promising tools to estimate machine utilization, especially if we consider the large potential of improvements that can be introduced on the neural networks that we built.

Chapter 4

Estimating Long Term Job Tardiness

The first part of this research dealt with designing neural network simulation metamodels to estimate long term mean machine utilizations given the job arrival rates and the dispatching rule applied in a job shop environment at two levels of complexity. In this second part of the research, the neural networks metamodels are applied to estimate long term mean job tardiness. The aim is to determine how well can we predict tardiness with neural networks and to determine which factors influence their predictive capabilities. This would allow investigating the effect of the performance measure being estimated.

As for mean machine utilization, we investigate two levels of system complexity which we will refer to as simple system case and complex system case. The experimental procedure that is followed in this section starts by designing metamodels from inputs generated from a wide range. If the neural network can predict performance (mean job tardiness) then complexity is added through the introduction of stochasticity. Otherwise, we split this wide range of the inputs to smaller subranges. Each resulting subrange would be analyzed similarly until the predicting performance improves or no improvement can be expected. This procedure allows us to examine the prediction performance of

neural networks with respect to the different system configurations and also allow us to assess the boundaries of the applicability of metamodeling with neural networks.

4.1 Case 1: Simple System

The system modeled in this first case is identical to the one used for the utilization measure in the previous chapter, simple system case (refer to Section 3.1.1). The only difference is due to the additional due date based job dispatching rules included for the tardiness measure.

4.1.1 Experimental settings

System Description

The changes made in this set of experiments, are related to the dispatching rules and to the introduction of a due date assignment method. For details regarding the original system, refer to Section 3.1.1. This system is allowed to operate with 3 dispatching rules: (SPT), EDD and MOD. The SPT rule, which is used also in the previous chapter, is introduced for the purpose of relating the two cases and the two due date based rules are used as they aim at minimizing tardiness, where one is a job based rule and the other is an operation based rule.

Because we want to measure tardiness, jobs need to be assigned due dates. The method used to assign due dates is the Total Work content method (TWK) and it involves a due date tightness factor, k , which is generated initially from the range $[2, 9]$. This range is selected according to pilot runs with different values of arrival rates and with different dispatching rules. These pilot runs indicate that the above range covers both the tight and the loose due dates. Hence, the due date tightness factor, k , is also presented to the neural network

as an input. In order to analyze the effect due date tightness factor, we further split this range into smaller ranges in our experiments.

As for mean machine utilization, in switching from deterministic to stochastic configurations, we apply the deterministic values as the mean of the corresponding exponential distribution both for the interarrival times and the processing times. Refer to Section 3.1.1 for further details.

Simulation Models

Different simulation models are developed in order to produce different sets of training and test examples. The models mainly describe the same system, however the range from which the input variables (in the data sets corresponding to each model) are generated, is varied from model to model. The ranges start from wide ranges to smaller ones in order to analyze the cases where the neural metamodel estimation is not acceptable. In the cases where neural metamodel perform well, these ranges are not split but they are used to generate the means of distributions for stochastic models. We investigate models with deterministic nature as well as models with stochastic interarrival times or with stochastic processing times or with both. Table 4.1.1 in Appendix B shows the list of the models. This table reports the nature and ranges of the inputs that are used in the training set and test set corresponding to each model. Figure 4.1.1 describes the relationship between the models. Table 4.1.2 in Appendix B shows the characteristics of the data sets generated from these models in terms the machine utilization as well as the proportion of tardy jobs (minimum, average, standard deviation and maximum).

Neural Networks Metamodels

The proposed neural networks are based on the Back propagation algorithm with two learning coefficients (the learning rate and the momentum coefficient). The size of the data sets (training and test sets) generated depends on the width

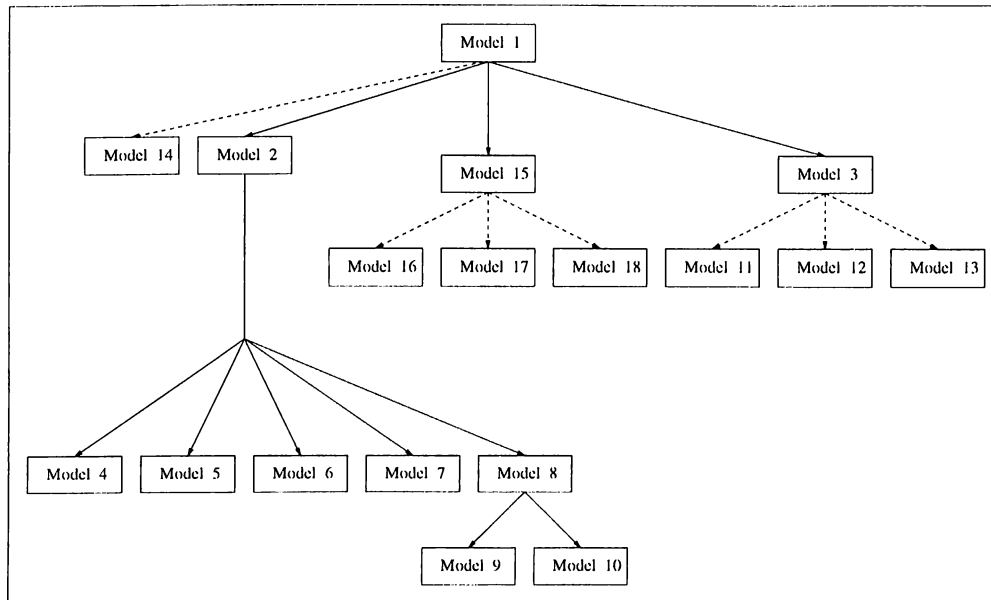


Figure 4.1: Mean Tardiness: Simple System: Relationship between models.

of the range of the corresponding inputs as well as on time constraints related to running the corresponding simulation programs and training the metamodels.

The generic network architecture that we use is as follows:

- Input layer: 5 processing units:
 - Interarrival time for job type 1, λ_1 .
 - Interarrival time for job type 2, λ_2 .
 - Interarrival time for job type 3, λ_3 .
 - Dispatching rule, ϕ : 1, 2 or 3 (SPT, EDD, MOD).
 - Due date tightness factor, k .

(For the stochastic case, λ_i stand for the mean of the exponential distribution modeling the interarrival time of jobs type i .)

- Hidden layers: Various combinations of networks with different number of hidden layers, with different number of processing units are used.

- Output layer : 3 processing units:
 - Mean job tardiness for jobs type 1,
 - Mean job tardiness for jobs type 2,
 - Mean job tardiness for jobs type 3.

Table 4.1.3 in Appendix B shows the list of the neural networks that were developed and their corresponding characteristics. In this table, the name assigned to each neural network are as follows: Exp3_A_B. This coding should be read as the name of the neural network number B developed as a metamodel for model A of this second experiment. An example would be: Exp3_4_3 . This describes the network number 3 designed for model 4 of experiment set number 3.

Error Assessment

Again, four measures are used to evaluate the performance of the neural networks. The first three are identical to the first three used in estimating mean machine utilization (Refer to Section 3.1.1). It is done so for the sake of consistency and to allow performing comparisons.

For the first measure (the tolerance approach), we use a tolerance level, α , that is allowed to take three possible values: 10, 20, 30 Min. Setting α at higher values will results in more precise networks but this will misleadingly improve the results. On the other hand, decreasing the value of α would result in more severe assessment. As a matter of fact setting an appropriate level of α for all the data set is difficult. Therefore, whatever way you look at it, this approach is subjective and so you cannot rely on it alone to assess the performance of neural networks. The MAD and MMAD approaches need some prior knowledge about the system and hence, require subjective interpretations. Thus, the first three methods will not be used to assess the acceptability of the performance of the metamodels but rather to compare the different networks constructed for each simulation model and to assess the relative effect

of the different factors considered. Method 4 (measuring percentage deviation from true value, simulation output) as described in Section 3.1.1, is no more applicable in this case because the true value (simulation output) may take zero values and hence cannot be used as a denominator. Instead, we propose a fifth error measurement approach for this experiment, as explained below.

We first define the following:

i : be the index that represents the example number in the training or test sets.

j : be the index to describe job type. $j = 1, 2, 3$.

S_{ij} : be the average tardiness obtained for job type j at example i obtained from the simulation model (true result).

N_{ij} : be the average tardiness obtained for job type j at example i obtained from the neural network metamodel (prediction of S_{ij}).

k_i : be the due date tightness factor used in example i .

TWC_j : be the total work content of jobs type j .

T : be the total number of examples in the training or test set.

We also define:

$D_{ij} = |S_{ij} - N_{ij}|$: absolute deviation per job type j and per example i ,

$D_i = \max\{D_{ij}, j = 1, 2, 3\}$: maximum deviation across jobs for example i ,

$F_{ij} = (k_i * TWC_j) + S_{ij}$: maximum of flow allowance and flow time for jobs type j at example i .

The fifth error assessment approach is described below.

- **Method 5: (deviation-to-flow time approach)**

The first three methods produced absolute measures of the error and hence are very subjective. The tolerance approach requires fixing some tolerance levels and these in turn requires an understanding and a subjective judgmental evaluation to set them. The MAD and MMAD approaches are measures related to the deviations in the estimate and hence, in the interpretation of the results, a prior knowledge of the system is needed. These methods will be then used just to discriminate between the different networks built. This fifth method provides a relative measure and consequently can be used to assess whether the networks achieved good results or not in terms of real life application. For each example in the data set, it provides a measure of the relative importance of the deviation between the simulation true results and the metamodel estimates with respect to the flow time of each job type. The interpretation of this measure is as follows:

Suppose that the neural metamodel is actually in use in a real system and that it is used to select an operational policy which is determined by a due date factor, k , and a dispatching rule, ϕ , as to meet management objectives in terms of job tardiness for a given combination of arrival times. The neural network then would be run several times to obtain the best operational policy, given the current job arrival rates. Now suppose that the selection process is over and that k and ϕ are decided upon. This means that management believes that the flow time that results from their decisions is acceptable. This value of the flow time would be equal to the sum of the flow allowance and the expected tardiness as estimated by the neural network. If the jobs appear to be more tardy than what was expected by the neural network, then it means that customers would have to wait more and so the metamodel must be penalized for this. If the job is tardy but less than estimated by the neural network, any plans such as transport of jobs would be disturbed. Thus, if the jobs are tardy, any deviation from the expected flow time as given by the metamodel should be penalized and thus, we measure the relative importance of the error in the estimate with respect to the actual flow time. If the job is not

tardy, then it means that the flow allowance given to each job is too large and so inventory costs would be incurred. Then we would like to know how important is the deviation from the flow allowance. Therefore, we measure the relative importance of the deviation from the true relative to the flow time, when the job is tardy, and relative to the flow allowance, when the job is early.

This approach takes the ratio of the deviation from the true to the flow allowance/flow time. If this ratio is low, then the performance is acceptable. Moreover, this approach has the advantage of evaluating the error associated with each example separately, hence it is objective as in the traditional percentage deviation approach (Method 4).

For each job type and for each example in the sets, the ratio of the deviation to the flow allowance is taken and then the average and the standard deviation of these ratios are considered across all the examples in the data set. Hence, the average error in each data set and for each job type, would be:

$$E_j = \frac{\sum_{i=1}^T \left(\frac{D_{ij}}{F_{ij}} \right)}{T}$$

We calculate the mean error and the corresponding standard deviation for each job type in order to see if the individual job characteristics can influence the performance of neural networks.

4.1.2 Results and Discussions

The detailed of the results of the metamodels for the four evaluation measures previously defined are provided in Table 4.1.4 in Appendix B. For each simulation model, we select the “best” neural metamodel (gray background) and we graph their relative performance in the graphs given in Figure 4.1.2 in Appendix A. By the term “best” we refer to the metamodel that outperforms the others for most of the error evaluation criteria (mainly on Method 5 as it is a more objective criterion). Except for Model 1, all the remaining 17 models

achieved good results with Method 5 (deviation-to-flow time method). Since we believe that this method is the most objective one and gives indications for the potential application in real life, we consider that -except for the very general case of Model 1- neural metamodels are able to predict mean job tardiness successfully (a mean deviation less than 6% of the flow time). Further fine tuning and experimentation with the neural network parameters as well as improvement in the learning algorithm will surely further improve the results. The next paragraphs will discuss the results in terms of error assessment methods 1, 2 and 3.

Model 1 is the general deterministic model. The results for this model are not acceptable by any of the error assessment methods. Models 2, 15 and 3 are special cases of Model 1 since we reduce the width of the interval from which job interarrival times are generated. They respectively correspond to the high, medium and low demand on system cases. Models 15 and 3 (medium and low demand on system) perform well in terms of all the error assessment methods. Thus, we develop three stochastic configurations from each of them: Models 16, 17, 18 from Model 15, and Models 11, 12, 13 from Model 3. These configurations are: stochastic interarrival times only, stochastic processing times only or both factors stochastic. For these stochastic configurations, the neural metamodels produced good results. Also, the deterioration in the performance of the metamodel relative to the deterministic cases is not significant. Comparing the stochastic configurations, it appears that there is no considerable difference between when the only stochastic factor is interarrival times or when the only stochastic factor is the processing times. However, when both stochastic variables exist, the results are slightly worse.

In Model 2, which corresponds to high demand on system, the neural networks performed poorly. From this, we develop Model 14 which is identical to Model 2 except that we remove all the examples in the data sets where the system is saturated (i.e. mean machine utilization is bigger or equal to 99%). This corresponds to removing the examples which have very low values of interarrival times. Model 14 achieved a considerable improvement over Model 2 for evaluation criteria 1, 2 and 3, although the results are not as good as the

ones achieved for the low and medium demand on system cases (Models 15 and 3). The results stated above indicate the demand on system (determined by the interarrival time) is an important factor with respect to the neural metamodel precision.

In order to see if any of the other factors (due date factor, k , or dispatching rule, ϕ) has an effect on neural network accuracy, we derive from Model 2, Models 4 through 8 in which we fix one of these two factor and we let the other to be variable. We also develop Models 9 and 10 where both the scheduling rule is fixed at MOD and the due date factor is set at 9 and 6, respectively. The improvement over Model 2 is not considerable. The reason for this may be that these two factors, when varied, do not result in a variation in the level of interactions that take place in the system. That is to say that, they act as to rearrange the flow of products in the system, but they do not change the volume of this flow. However, there appear to be a variation in the metamodel precision depending on which factor is fixed and at which level. This variation is also dependent on the error assessment method. This inconsistency between the evaluation criteria does not allow us to make any conclusion regarding the exact effect of these two factors.

If we consider error assessment Method 5, the neural metamodel performed better on job type 3 than on job type 2 and job type 2 better than on job type 1. This ordering according to the job types indicates that the job characteristics play some role in the accuracy of the metamodel. This ordering corresponds to the number of operations of each job type.

The error measurement method is an essential issue to be decided upon for two main reasons: the first reason is that the ranking of the performance of neural networks may vary according to these methods for some cases. In that case, no absolute conclusion can be made with respect to which metamodel is better than the others as the order of models changes from error assessment method to another. The second reason is that in cases like this experiment -where the usual relative measure cannot be used- evaluating the performance of our networks is a subjective procedure. This evaluation depends on the

real users of the metamodel. Therefore, both metamodel developers and users have to agree upon an evaluation procedure before starting the metamodeling process.

Demand on system is an essential factor. If system is heavily loaded, the predictive capabilities decrease. This is the same as to say that high load results in more complex interactions within the system and so more difficulties in classifying those interactions.

Due date tightness factor is not as considerable as the demand on system factor. However, reducing the range of those two factors does contribute to improving slightly the precision of our estimations.

The scheduling rule being used at tight due dates may have an important effect. Further analysis and experimentation is required to investigate this point.

Allowing the system to handle stochastic configurations when the demand on system is not high, does not seem to deteriorate the results considerably. This may be due to the fact that we are assessing the performances in terms of averages. In these stochastic models, the standard deviations related to the errors of the networks slightly increased with respect to the deterministic case. However, these standard deviations are still small enough to allow us to say that the network results are reliable (i.e. we can be quite confident that the result is not too far from the true performance).

For this simple system case, the metamodel generalization capabilities were good and this enhances the applicability in real life.

4.2 Case 2: Complex System

4.2.1 Experimental settings

System description

The system examined in this second case for estimating job mean tardiness, is identical to the one used for the second case of estimating mean machine utilization (refer to Section 3.2.1. for more details)

Simulation Models

For this experiment, 8 simulation models are built. Table 4.2.1 in Appendix B shows the list of the models. This table also reports the nature and ranges of the inputs that are used in the training and test sets corresponding to each model. Figure 4.2.1 describes the relationship between the models. Table 4.2.2 in Appendix B shows the characteristics of the data sets generated from these models in terms the machine utilization as well as the proportion of tardy jobs (minimum, average, standard deviation and maximum). This study is different from the one made in the previous case: we have previously shown that the most considerable factor on neural network metamodel's predictive capabilities is the factor of demand on system. Hence, in this study we only focus on this factor. For the simulation run lengths and transient period, they are identical to the corresponding case in estimating mean machine utilization (refer to Section 3.2.1 for further details).

Neural Network Metamodels

For each model described in the previous section, three neural metamodels are built. The network learning rate and momentum term are set fixed and the architecture is varied in terms of number of hidden layers and number of

processing units per hidden layer. The number of processing units in the input layer and the output layer as well as the in the hidden layers are increased to adapt to the increase in the system size and to the number of job types, relative to the simple system case. Through the cases examined so far, we tried to keep the same type of neural network architectures with the same values of network learning rates and momentum terms. This is necessary in order to be consistent in the comparisons between the different cases. The number of training cycles is set to a large value to total to 500000 examples presented to the neural network in the training phase. This allows to all the network to converge.

The generic network architecture that we use, is as follows:

- Input layer: 8 processing units:
 - Interarrival time for job type 1, λ_1 .
 - Interarrival time for job type 2, λ_2 .
 - Interarrival time for job type 3, λ_3 .
 - Interarrival time for job type 4, λ_4 .
 - Interarrival time for job type 5, λ_5 .
 - Interarrival time for job type 6, λ_6 .
 - Dispatching rule, ϕ :1, 2 or 3 (SPT, EDD, MOD).
 - Due date tightness factor, k .

(For the stochastic case, λ_i stands for the mean of the exponential distribution modeling the interarrival time of jobs type i .)

- Hidden layers: For each model three network architectures are applied with different number of hidden layers and with different number processing.
- Output layer : 6 processing units:

- Mean job tardiness for jobs type 1,
- Mean job tardiness for jobs type 2,
- Mean job tardiness for jobs type 3.
- Mean job tardiness for jobs type 4,
- Mean job tardiness for jobs type 5,
- Mean job tardiness for jobs type 6.

Table 4.2.3 in Appendix B in shows the list of the neural networks that were developed and their corresponding characteristics.

Error Assessment

For the sake of consistency, the same four error assessment approaches that are used in the previous simple system case, are used in the complex system case also. Refer to Section 4.1.1 and to Section 3.1.1 for more details.

4.2.2 Results and Discussions

Now, we investigate the use of neural networks for predicting mean job tardiness in a more complex job shop system. The detailed performances of all the designed metamodels across the four evaluation method previously defined are provided in Table 4.2.4 in Appendix B. For each simulation model, we select the “best” neural metamodel (gray background) and we graph their relative performance in the graphs given in Figures 4.2.1 in Appendix A. By the term best we refer to the metamodel that out performs the other on most of the evaluation criteria (mainly the fourth as it is a more objective criterion).

Model 1 is the general case. The performance of this model is not satisfactory for all the error assessment criteria. In the simple system case, we observe that the metamodel precision is mostly determined by the demand (or load) on system, and hence we check if this factor still has the same effect

when the system complexity is increased. Thus, we develop Models 2, 3 and 4 which correspond to high, medium and low demand on system respectively. Model 2 (high demand on system) poorly performs on all the criteria where as Models 3 and 4 perform well. This confirms our previous observation. We also develop Model 5 which is identical to Model 1, except that we remove all the examples in the data set for which the system is saturated (i.e., where the mean machine utilization is larger than or equal to 98%). This model achieved much better results than Models 1 and 2, further confirming our previous observations. Specifically, this model shows that as we increase the demand on system, estimation of job tardiness becomes more difficult due to the increase in the interactions in the system. According to error assessment Method 5, Model 5 achieves acceptable results. The models discussed so far, indicate that the neural metamodel accuracy does not deteriorate linearly with the increased demand on system: This means that when moving from low demand to medium, the deterioration is insignificant; and when moving from medium to high demand (but the system is still not saturated), the deterioration is slightly more important. However when moving from high demand (with system not saturated) to high demand (with system allowed to be saturated) the deterioration is very important. This exponential pattern in the deterioration of the accuracy of the metamodel, where most of the deterioration takes place at the extreme case only, indicates that the potential of applicability of neural metamodels in real life is high.

Since Models 3 and 4 (medium and low demand on system), achieved acceptable results, we investigate the corresponding stochastic configurations for the combination of these two models. This results in Models 6, 7 and 8. The results achieved by these models are very much comparable to the results achieved by Model 3 (deterministic, medium demand), indicating that the introduced stochasticity does not significantly affect neural network predictive capabilities. It also appears that neither the nature of the stochastic factors, nor the number of stochastic factors have an important effect on the predictive performance of neural networks.

Another observation is that if we look at the fifth error assessment approach,

the neural metamodel perform better on job type 6 than on job type 5, than on job 4, and so on, until job type 1. This ordering according to the job types indicates that the job characteristics play some role in the accuracy of the metamodel. In fact, this ordering is inversely proportional to the number of operations of each job type. This is a strange behavior as we would have expected the inverse to take place.

Regarding the generalization capabilities of neural networks, we cannot conclude as it depends on the error assessment criteria. The generalization capability is better when we consider the MAD or the MMAD approaches than when we consider the tolerance approach or the flow time approach. Despite this, the difference between the performance of the training sets and the test sets on the latter criteria, is not large enough to threaten the applicability in real life.

Some pilot experiments were done on using neural metamodels to predict mean job flow time. These experiments gave some indications that estimating mean job flow times is very similar to estimating mean job tardiness. However, detailed analysis is required in order to confirm this observation.

4.3 Comparison of Simple & Complex systems

Unlike the mean machine utilization case, where in both simple system and complex system case, the neural networks metamodels perform well at all the models investigated, in the mean job tardiness case, there appears to be some important changes as we move from a simple to a complex case.

In both the simple system and complex system, demand on system (determined by the interarrival times) is a considerable factor in affecting neural metamodel accuracy. Experiments have also shown that neural metamodel produce good results at low and medium demand levels, for the

deterministic and stochastic configurations, and both for simple and complex systems. However, when the system is highly loaded (with system allowed to be saturated), neural metamodels performed acceptably (according to error assessment Method 5) for the simple system case and performed poorly in the complex system case. If we consider the high demand models, where the system is not allowed to be saturated, then performance is acceptable for both simple and complex systems.

The experiments have also shown that there is an exponential behavior to the deterioration of the accuracy, with most of the deterioration occurring in the very high demand case. This fact is highlighted by the first graph on Figure 4.3. Figure 4.3 is in terms of the third method of error assessment (MAD). This result indicates that metamodels are able to predict mean job tardiness for systems in steady state successfully, although the users should expect some deterioration in the performance as we move to more complex systems and as we move to more loaded systems.

In terms of the effect of stochasticity, we observe that increased system complexity has a negative affect on metamodel performance. However, this negative effect is not important enough to reduce the potential of applicability of neural metamodels in real life. Also, this negative effect does not depend on the nature of the factors being stochastic but much more on the number of stochastic factors for both simple and complex systems (within the stochastic factors tested) as shown by the second graph on Figure 4.3.

In both simple and complex systems, we observe that according to the fifth error assessment approach, there is an ordering of the performance of the metamodels according to the job type and this ordering is consistent through the different models. In the simple system case, metamodel accuracy is better more jobs with more operations (Figure 4.1.2). For the complex system case, jobs with more operations achieved better estimates (Figure 4.2.1). While the ordering in the simple system is expected, it is not the case for the complex system. In fact, when a job has more operations, it means that it would interact with more other jobs and hence, its behavior is more difficult to estimate.

The error assessment methods did not produce very consistent results in both cases in the sense that we would consider the metamodel performance acceptable on some method, but not on another. This highlights the fact that the first three error assessment criteria are subjective and hence we cannot rely on them to make judgments. But rather, we base our assessment on Method 5.

The third graph and fourth graph on Figure 4.3 show that system complexity, demand on system and stochasticity have a negative effect on generalization capabilities. In fact, generalization capability deteriorates more as we move from low demand to higher demand with most of the deterioration taking place at the extreme. In the third graph of Figure 4.3, the network seems to perform better at very high demand than at high demand (complex system). This is not the case because the point shown on the graph corresponds to a network trained on much larger range of inputs which explains that abnormality. In addition, it appears that generalization capability is not affected by the nature of the factor being stochastic but more by the number of stochastic factors, where this observation is valid at both levels of complexity.

Chapter 5

Estimating Short Term Job Tardiness

In the previous two chapters, we measured the performance of neural networks as a simulation metamodeling tool for estimating long term or steady state system performance. In this chapter we investigate the use of neural network simulation metamodels to estimate short term system performance. We only focus on job tardiness as it is representative of the performance measures considered by modern management in selecting the appropriate operational policies. Here we didn't consider machine utilization as another measure, because machine utilization is more relevant in long term decision making such as capacity planning than in short term decision making. Since we deal with short term system performance, the initial system status becomes a relevant factor in this analysis, in addition to the factors considered for the long term system performance. In this part of the research, mainly the simple system case is investigated. We try one complex model which indicated that the error is too high as expected. Thus, we did not continue investigating this part as that would be a replication of what takes place for the simple system. We focus the analysis on the effect of starting system condition as well as on the effect of demand (or load) on system (determined by the arrival rates).

5.1 Experimental settings

System Description

For this set of experiments, we consider the same simple system which was used in the last two previous chapters. This system consists of 4 machines, 3 product types and 3 transporters. It operates under 3 job dispatching rules: (SPT), (EDD) and (MOD). Due dates are assigned according to the Total Work content method (TWK). In the experiments, we consider the system configuration where both processing times and interarrival rates are stochastic. Our pilot runs have indicated that stochasticity has an important effect on system performance when the simulation run is short. As a result, we consider the most difficult configuration (relative to the previous two chapters). Moreover, the previous experiments have revealed that stochasticity reduces the precision of metamodels but does not alter the impact of the factors investigated. In order to keep a basis for comparisons, we use exactly the same system characteristics and parameters as previously explained in Section 4.1.1.

Simulation Models

For this set of experiments, 26 different models are built in order to investigate the effects of initial system status and the effect of system load. Model 1 corresponds to a deterministic configuration. The next 24 models correspond to the stochastic interarrivals and stochastic processing time configuration. The last model corresponds to the deterministic complex system.

We define the initial system status in terms of the number of each job type waiting in the input queue of each machine. Another approach would have been to just specify the number of each job type present in the system as a whole and not specifying at which level of processing they are or to consider only jobs at the start of their routings (omitting any work in progress inventories). By that way, we think that we fully and realistically describe the initial status of the

system. Since we allow the system to initially have work-in-progress inventory, the due date assigned to these jobs is proportional to their remaining total processing time (similar to newly arriving jobs whose due dates are proportional to their total processing time which is equal, at their release time, to the remaining total processing time).

Another important issue when dealing with short term performance is the simulation run length. Very short runs usually result in low number of observations. On the other hand, long runs imply that the system can reach the steady state and hence, the effect of the initial status cannot be seen. In our previous experiments on the long term system performance, we consider a transient period of 2 work days during which statistics are cleared. This period is selected based on some pilot runs. In this set of experiments on short term performance, we consider a simulation run length of 2 work days, that is the same transient period. Table 5.1 in Appendix B shows the list of the models built. This table also reports the ranges from which the interarrival times, due dates tightness factor and initial number of jobs at each machine are selected. Figure 5.1 describes the relationship between the models investigated. Table 5.2 in Appendix B lists the characteristics of the data sets generated from each model in terms of machine utilization, proportion of tardy jobs and tardiness.

Neural Network Metamodels

As in the previous experiments, the proposed neural networks are based on the Back propagation algorithm. Regarding the size of the data sets, we are able to use large sets, since the simulation run lengths are short. The training sets contain 600 examples where as the test sets contain 350 examples. For each model described in the previous paragraph, we built two neural network metamodels: one with a single hidden layer and one with multiple hidden layers, with number of processing units: 15-20-25-20-3 and 15-45-3, respectively. The learning parameters is set at 0.9 and the momentum term at 0.6. Moreover, we increase the size of the input layer in order to include the information regarding the initial status of the system. In addition, we

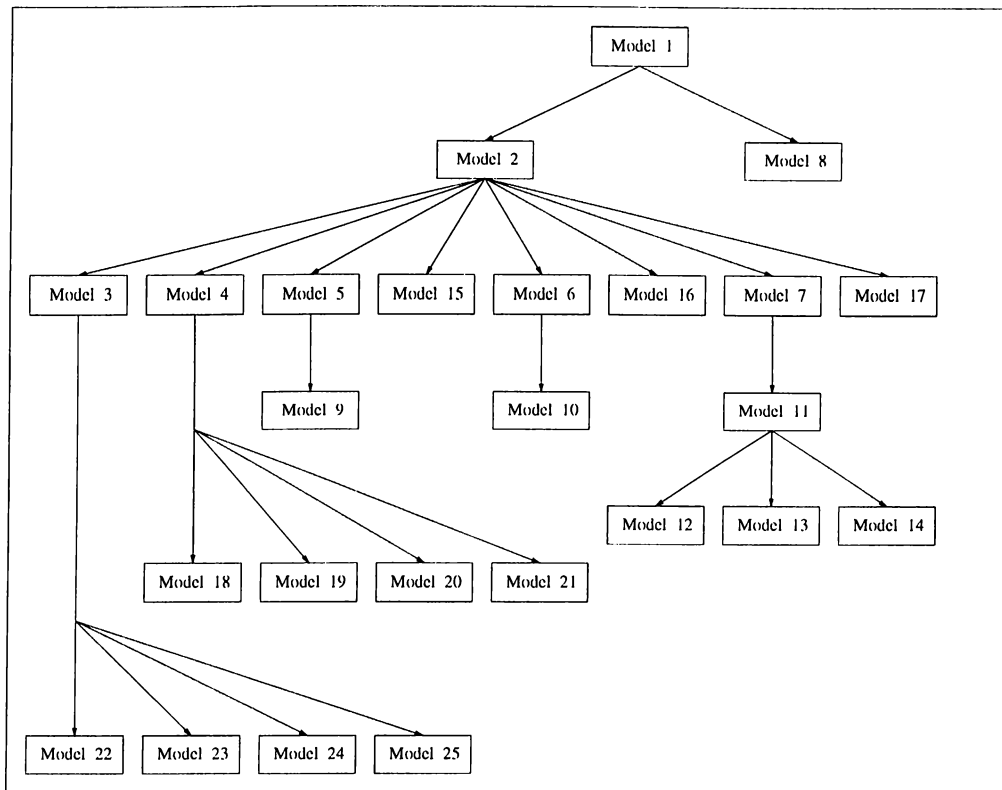


Figure 5.1: Mean Tardiness: Simple System: Short term performance estimation: Relationship between Metamodels.

build larger networks (in terms of the number of processing units per layer) as compared to the previous chapters. This is because we expect that transient state behavior is more difficult to estimate than steady state behavior.

The following generic architecture is used:

- Input layer: 15 processing units:
 - Interarrival time for job type 1, λ_1 .
 - Interarrival time for job type 2, λ_2 .
 - Interarrival time for job type 3, λ_3 .
 - Dispatching rule, ϕ : 1 or 2 (SPT,FCFS).
 - Due date tightness factor: k .
 - Number of jobs of each type at machine j (10 nodes).

- Hidden layers: single layer and multiple layer.
- Output layer : 3 processing units:
 - Mean job tardiness for jobs type 1,
 - Mean job tardiness for jobs type 3,
 - Mean job tardiness for jobs type 3,

Error Assessment

To be consistent with the last chapter on estimating long term job tardiness, the same four error assessment methods that are applied for this set of experiments. Namely, these are the tolerance approach, MAD, MMAD and the flow time approach. The reader can refer to Section 4.1.1 for further details.

5.2 Results and Discussions

The results are presented in Table 5.3 in Appendix B. The graphs of these experiments are depicted in Figure 5.2 in Appendix A. The results are less consistent than in the previous two experiments on long term system performance.

Let us first consider the error assessment Method 5 (flow time approach). The performance of the neural networks on the test sets is consistent with its performance on the training set. Model 1 is the only deterministic model, the remaining models correspond to the stochastic configuration of the system. Model 1 is comparable (general deterministic model) with Model 2 (general stochastic model), we observe that stochasticity has a negative effect on metamodel performance. This corresponds to the conclusion made in the previous chapter. The performance of the metamodel constructed for Model 1 is about 7 % on the training set and about 13 % on the test set. For Model 2, the metamodel achieved about 8 % on the training set and about 23 % on the

test set. This shows that in terms of generalization capability (which is the bottom line), stochasticity has an important negative effect. This observation is in line with our previous finding in the last chapter.

Models 2 through 25 correspond to the stochastic configuration. If we compare Model 2 (tight due dates) to Model 8 (loose due dates) we observe that there is no important difference. Hence, due date tightness factor has no effect. Models 9 through 14 have a fixed initial number of jobs at each machine for each job type. For these models, we observe that increasing the initial number of jobs results in improving the results until some point then no improvement is observed. Even if this initial number exceeds the average number of jobs in each queue (at steady state), metamodel performance does not change. The initial improvement could be explained by the fact that the starting system state tends more towards its steady state. The lack of improvement that takes place after some point could be explained as follows: In line with our previous finding in the last chapters, we noted that when the system is highly loaded, the network performance deteriorates. Thus, the improvements achieved by the fact that the system tends towards the steady state is canceled out when the system goes beyond its steady state (and becomes highly loaded). The pattern of behavior just mentioned is also repeated when we split the range of the interarrival times to form models 18 to 21 and models 22 to 25 corresponding to low demand and to high demand, respectively and hence confirming our observation.

Another observation is concerned with the effect of demand (or load) on system. Comparing Model 3 to Model 4 (high to medium demand) and Models 18 through 21 to models 22 through 25 (medium to high demand), we observe that the demand on system has an opposite effect than what was observed in the previous chapters. In the previous chapter we noted that the increasing load on system results in decreasing the metamodel accuracy in estimating long term job tardiness. In this case, we observe the opposite, that is the higher the demand, the better is the metamodel accuracy. The reason for this is that the higher the demand, the quicker the system will reach its steady state and the more stable it will be, and hence, the more regular the system behavior will

be.

Regarding the generalization capability, the results show that the size of the range from which the data is generated plays an important role (while keeping the same number of examples generated from that range). Whenever this size is reduced, the better is the generalization capability (i.e. the performance on the test set is nearer to the performance on the training set). The generalization capability is not considerably affected by the load in system as was the case in the previous chapters. The reason for this may be due to the introduced stochasticity (stochastic interarrival times and stochastic processing times). In short term, system behavior is very sensitive to stochasticity and thus a very large variety of behaviors exists. Hence, by reducing the range from which training examples are generated and keeping the same number of examples, we are providing a better coverage of the system behavior that takes place in that range and consequently the network accuracy improves. This observation did not come out when we estimated long term job tardiness since the system was in steady state and its behavior was regular.

In addition, we note that in the models where the initial system status is selected from an interval (models 15, 16 and 17) and in the models where the initial system status is fixed at the middle point of this interval (models 5, 6 and 7), the networks performed comparably. This observation indicates that reducing the range does not contribute to improving the metamodel accuracy considerably but does affect the generalization capabilities as previously mentioned. These two observations indicate that there is some trade off to be taken: Either the user builds multiple metamodels to gain in generalization capabilities (one metamodel corresponding to one subset of the interval from which the inputs are generated) and loses in metamodel development time, or the opposite. We believe that the second alternative is better if we consider that the metamodel accuracy can still be improved and that the results will still be validated with the simulation model.

Concerning the results themselves, which is the bottom line, the experiments show that the higher the demand on system and the more loaded is the

initial system status, the better the results. The accuracy of the metamodel at low demand and empty initial status is not acceptable (error more than 10%). However, given the lack of emphasis on fine tuning the network architecture much better results could be achieved.

When we consider the second and the third error assessment methods (MAD and MMAD), we see that the average deviation in the estimate is always proportional to the average tardiness (deviation in the range of 5 % to 30 % of the average tardiness). We cannot interpret these results in terms of their acceptability or not as this depends on the application. In terms of comparing the different models, the observations made previously, regarding the effect of initial system conditions and system load, for the fifth error assessment method also hold here.

When using the first error assessment method (tolerance method), the results are mixed and somehow confusing. In fact, we can observe some inconsistencies between the ranking of the models relative to the training sets and tests sets. This behavior is difficult to explain. However, the trends previously mentioned do also hold here. That is to say: higher load on system improves the accuracy of the metamodels and also higher initial queue sizes improves the results until some point than no improvement is observed.

The graphs on Figure 5.3 in Appendix A, show the effect of introducing stochasticity or system complexity relative the initial deterministic configuration, with respect to the training set, test set and to generalization (defined as the difference between them). The complex system used is the same as the one used in the previous chapter. The three graphs shown correspond to error assessment Methods 2, 3 and 5. We don't include Method 1 because it does not highlight well the difference between these cases. The three graphs on Figure 5.3 of Appendix A are consistent in showing that the effect of system complexity is far more significant than that of stochasticity in deteriorating the metamodel performance. Moreover, these graphs indicate that generalization capability suffers from stochasticity and system complexity. Another observation is that according the Method 5, stochasticity has an insignificant effect compared to

the effect of complexity, yet, the effect of stochasticity is important. This observation is in line with our previous expectations and in line with our findings in chapter 4 (estimating long term job tardiness). This may lead us to intuitively expect that the patterns of behavior we have observed for the stochastic system (as described before) will still be valid if we also analyze the complex system case in detail. However, this cannot be done until we construct metamodels for the stochastic case that would produce highly acceptable results.

In conclusion, we can say that in estimating short term system performance both system initial condition and demand on system considerably influence the performances of the metamodels. Decreasing the size of the range of the inputs does not improve the accuracy on the training set but improves the generalization capabilities. Moreover, stochasticity and system complexity have a negative effect on network accuracy and on its generalization capability, where the effect of system complexity is by far more significant. A last remark is that the results are not very good for some of the models but they are not bad enough in order to assert that neural networks are not appropriate for estimating short term system performance. On the contrary, we believe that with further fine tuning of the network architecture and parameters, the network could achieve much better results.

The purpose of using neural metamodels to estimate short term tardiness, was to investigate their potential application in real time decision support. For this purpose, we include one last test. We would like to know if by using the neural metamodel, we are able to make the best decision. In other words, we would like to know if, based on the metamodel, we would select the same dispatching rule as we would do if base the decision on the simulation model. This test is applied on three general models, corresponding to deterministic simple system, stochastic simple system and deterministic complex system. In selecting the dispatching rule (based on the simulation models or on the neural metamodel), several alternatives maybe available. If the job types have different weights, then we may select the rule that minimizes the tardiness of the tardiness of the job type with highest weight. This may be the case

when one job type is very profitable or very strategic to the business. Another alternative would be to select the rule that produces the minimum overall average tardiness (across all the job types). The following table show the results of this final test, where the columns correspond to different decision basis (based on each of the existing job types or on the overall average).

System	Job type 1	Job type 2	Job type 3	Job type 4	Job type 5	Job type 6	Average
Simple, Deterministic	81%	100%	79%	NA	NA	NA	79%
Simple, Stochastic	96%	84%	49%	NA	NA	NA	91%
Complex, Deterministic	100%	100%	71%	97%	98%	100%	100%

NA: Not applicable

Testing capability to select operational policies.

The results of this test are unexpected. Recall that from Figure 5.3, we observed that stochasticity and system complexity have a negative effect on the accuracy of the metamodel. The above table gives opposite indications. If the selection is based on the overall average tardiness, the metamodel on the complex deterministic system ranks better than on the simple stochastic system, which in turn ranks better than on the simple deterministic system. This is exactly the inverse of what Figure 5.3 indicated. The reason for this contradiction may be that complexity (whether due to stochasticity or to system) acts as to increase the difference between the performance of the different dispatching rules. Thus, it is easier for the metamodel to reflect this. In fact, this test implies that using neural metamodel is likely to produce good decisions as the system gets more complex and hence, neural metamodels are likely to be included in real time decision support systems. In addition, the test shows that making the decisions based on different job types leads to different metamodel performance. This is in line with our previous observation, related to the use of error assessment Method 5 (deviation to flow time approach). This method has indicated that metamodel performance is related to the job type characteristics. This test also supports our previous argumentation regarding assessing the validity of the metamodel. This test illustrates that different criteria lead to opposite interpretations or different decisions. Hence, the choice of the criteria should depend on the objective of the study.

Computational requirements

At this point, it is also important to mention the computational burden associated with this type of experimental investigation. In this research, the results of 64 simulation models and 193 neural networks are reported. The time required to develop the training and test sets depends on the number of examples, in the data set, the size of the system being modeled, the simulation run length and the frequency of events taking place in each example of the set (the number of events is a function of the arrival rates and of the processing times). For example, consider the case of the general simple deterministic system built to estimate long term system performance. In order to generate 100 examples from this model, it would take about 60 minutes of CPU time on a SUN SPARC 2 station. This figure could be much less if the examples correspond to a low loaded system and much higher for a highly loaded system. In the experiments carried out, a minimum of 600 and a maximum of 1000 examples are generated from each simulation model. Thus, on the average the computer usage per model is 9 to 10 hours of CPU time. As for the simulation models, the length of the neural network session is variable. In the networks built, we fix this length to 500000 examples presented to the network. This is a large number but we did so in order to ensure convergence. The time required by the network to train on these examples, depends on the network architecture used: the more processing units and/or hidden layers there are, the longer is the training session. Larger networks are employed when metamodeling complex systems and when metamodeling short term estimation. Serial PCs are used to simulate the parallel processing that takes place in the network. An average training session would take on 486 IBM compatible PC, 50 MHz about 3 hour. The recall session of the neural metamodel (requiring the network to generate outputs for examples not included in the training set for the purpose of testing its performance) is a very fast process and it would need about five seconds to recall all the examples in the largest set. In fact, this was one of the advantages of neural networks once trained. Given that about three neural networks are built for each simulation metamodel, then a total of about 18 hours were required for each of the 64 models examined. Thus, the total volume

of computer usage of all this experimental investigation would be about 1152 hours of CPU time. These figures do not include the time required to develop and validate the simulation models, nor the post-data processing that takes place to assess metamodel accuracy. The reason these number are provided to the reader, is to show that the development of neural network metamodel is a long process and is consuming in terms of time and computer resources. The burden of this off-line work can be recovered by the savings achieved in the on-line application of the metamodel. One last word is that the continuous innovations in the hardware (even the serial one) are reducing the development time significantly.

Chapter 6

Concluding Remarks & Future Research Directions

In the first chapter we lay down the background of this research. It appears that research in approximate techniques such as simulation metamodels is maturing, reflecting a need by modern business for fast approaches to help solve operational problems. Research on the use of neural networks as a simulation metamodeling technique is not abundant in for estimating long term system performance and rare for estimating short term system performance. This work aims at enriching this literature.

This research also has investigated the effect of several factors on the predictive capabilities of back propagation neural networks applied on job shop system. These factors are:

- * Study horizon: Short term vs. Long term
- * Performance measure to be estimated: Mean machine utilization vs. Mean job tardiness.
- * System complexity: Simple system vs. Complex system.
- * System configuration: Deterministic vs. Stochastic (3 cases).

- * Error assessment methods: Relative vs. absolute measure.
- * Metamodel design: Single general metamodel vs. multiple specific metamodels (through reducing the range from which the of variables in data sets are generated).
- * Neural Network architecture: Single hidden layer vs. multiple hidden layers.

The experiments carried out in this research have shown that neural networks are very a promising tool as a simulation metamodeling approach. Despite the fact that our emphasis was not on developing the most accurate metamodels, but rather our emphasis was to compare the effect of the factors mentioned above, the neural metamodels have achieved an acceptable level of accuracy. Given that the neural network design parameter ranges were only partially covered and the learning algorithm could be much further improved, we believe that neural metamodels are able to achieve better results than the ones reported in this research.

With regard to our experiments on estimating long term system performance, we come with some interesting observations as follows:

1. Our experiments have shown that neural network metamodels accuracy is affected by the performance measure in use (mean machine utilization or mean job tardiness). First, the metamodels investigated performed better for estimating machine utilization than job tardiness. The reason for this is that machine utilization is mostly determined by the arrival frequency to the system and processing times. On the other hand, job tardiness is determined by the arrival rates, by the due date assignment procedure, by the dispatching rule and by the processing times. Hence, tardiness is a more involved function to estimate. Some pilot runs indicate that by using neural metamodels to estimate mean job flow time, the results obtained are very similar to estimating mean job tardiness, however, thorough analysis is required to confirm this.

2. In our experiments, we define system complexity in terms of increased system size and in terms of increased number of job types flowing through the system. System complexity was shown to affect negatively the performance of neural metamodels, where this negative effect appears more significant on the mean job tardiness than on the mean machine utilization. Despite this, the deterioration is not considerable enough to completely threaten the application of neural metamodels in real life. The threat takes place when the system is subject to extremely high demand on system (in cases where the system is not in steady state).
3. Stochasticity is another dimension of complexity which was investigated for the cases of medium and low demand on system. In these cases, the experiments have shown that stochasticity reduces slightly metamodel accuracy, but still within very acceptable limits. In addition, it appears that the nature of the factors investigated (interarrival times or processing times) is not important, but rather it is the number of factors that are stochastic that is important, for both performance measures. Moreover, this negative effect is not exaggerated by the effect of system complexity. Hence, at medium or low demand on system, and at simple or complex systems, and at mean utilization or mean tardiness, stochasticity can be handled successfully by neural metamodels.
4. The error assessment approach is a very critical issue. First, we distinguish two different uses of the error assessment: The error assessment approach, when used to study the significance of the effect of different factors can be allowed to be subjective. This is because the values produced by each method on one factor is interpreted in terms of the other values on the other factors. However, when it comes to assessing the acceptability of the results, subjective error assessment methods can no more be relied on. We base this type of assessment on objective methods, which may be subject to discussion as this depends on the final application of the metamodel. The second point related to the error assessment approach, is that on the overall they were quite consistent (i.e. metamodels ranked similarly on the different methods). This consistency

is a confirmation of the conclusions made. However, the significance of the effects that each factor investigated on the metamodel performance changes as we move from subjective to objective approach (in the case of mean job tardiness). In summary we say that in the analysis of the literature on neural metamodels, one should always motivate the error assessment approach applied relative to the final application of the metamodel, otherwise the validity of the results achieved can always be questioned.

5. Both for mean machine utilization and for mean job tardiness, the demand on system (determined by the arrival rates) appears to be a very considerable factor relative to metamodel accuracy. As a matter of fact, the higher the demand the lower the accuracy. We observe that deterioration of accuracy relative to demand on system follows an exponential pattern with most of the deterioration taking place at the extreme case (the case where the system is out of steady state or nearly). This pattern of behavior indicates that if we slightly move away from the saturation case, say with a maximum utilization of 90%, we are able to model reasonably loaded systems with a well acceptable level of accuracy. Also because of this dependence on the demand on system, it is wise to construct neural metamodels for every range of demand on system as we did in the experiments as this way would distinguish between extremely reliable metamodels and others less reliable.
6. The experiments with mean job tardiness with simple systems indicate that the due date tightness and dispatching rules have some influence on the accuracy of the metamodel. However, the error assessment approach results are not consistent enough on this point to allow us to make conclusions regarding which factor is more important. The only observation that can be made is that they are much less considerable than the demand on system.
7. It appears that in estimating mean tardiness, neural metamodels perform differently for each job type. The results show that the accuracy is proportional to the number of operations of the corresponding job type:

the less operations a job has, the less accurate is the estimate of the job tardiness and vice versa. This observation is unexpected and the experiments done so far do not allow us to give a reasonable explanation.

8. Generalization capabilities are quite good for neural network metamodels, although these capabilities are negatively affected by (in order of importance) increased system complexity, by increased demand on system as well as by introduced stochasticity. This is very important point since the bottom line is that in real life applications, we would rely on the generalization capabilities of the metamodels to make the estimations and they must be as trustful as on the training sets.
9. Regarding the neural network design, unfortunately, no architecture is always better and no values of the network parameters (learning rate and momentum term) are good for all the models. In addition to this, we do not have an indication about the size of the data sets that should be used nor on the length of the training period. This led us to try and take the safe way, within the available time constraints and to increase the size of the data sets and the length of the training sessions. This resulted in long metamodel development phases. Some work has still to be done on these issues.

With regard to our experiments on estimating short term system performance, we can make the following observations:

10. Stochasticity has an important impact on metamodel accuracy but, this impact is far less important than that of system complexity.
11. independently of whether we are estimating short term or long term job tardiness, stochasticity and system complexity keep the same effect on the accuracy of the metamodels.
12. Generalization capabilities suffer also from introducing stochasticity or system complexity

13. In estimating short term system performance, the demand on system still plays an important role in determining the accuracy of the metamodel. On the contrary of estimating long term performance, increasing demand on system results in improvement of the metamodel accuracy which can be explained by the fact that increasing system performance results in the system tending quicker to its steady state and hence tending more to have regular pattern of behavior.
14. Initial system status appears to have a considerable impact on metamodel accuracy until some point and from that point on, this impact vanishes.
15. Decreasing the size of the range of the inputs does not improve the accuracy on the training set but improves the generalization capabilities. Also the generalization capability is not considerably affected by the demand in system as was the case in the previous chapters.

In the second chapter of this manuscript dealing with laying the research background, we present a set of issues that needed to be clarified and investigated in order to assess the potential of applicability of neural networks as a simulation metamodeling tool. In this thesis, we tried to get into some of these issues and we came up with some interesting observations. We tried to highlight some of the factor that influence neural metamodel accuracy and we showed that neural network is a promising approach for simulation metamodeling in estimating long term system performance. We also investigated the possible use of neural networks to estimate short term system performance. Our results show that such an application may be possible which implies that neural networks have the possibility to be used for real time decision support. Given the computational requirements of such a study, we did not give much emphasis on fine tuning the metamodels, thus the results achieved could be further improved. Our results also indicate that system regularity is a key issue for the accuracy of the metamodel. Whether we are estimating long or short term performance, it is better to model a system in steady state and which has a regular behavior. Moreover, the experiments show that the use of neural metamodels in real life is not straight forward as

the reported successful case studies may lead us to think. In fact, stochasticity and especially system complexity have an important impact on metamodel performance.

In order to further investigate the potential of neural networks as a simulation metamodeling technique and possibly enhance its applicability, we suggest the following research directions:

- Investigating the effect of system configuration: Flow shop to job shop
- Investigating the effect of system disturbances such as machine break down
- Investigating of applying different distributions in studying stochasticity
- Investigating the effect of the size of data sets.
- Investigating the effect of estimating other measures such as estimating standard deviation of the performance measure.
- Developing rules or guidelines as of how to select the best network architectures
- Investigating the validity of neural metamodels with respect to real systems

Bibliography

- [1] Kenneth R. Baker. Sequencing rules and due date assignments in a job shop. *Management Science*, 30(9):1093–1104, september 1984.
- [2] Russell R. Barton. Metamodeling for simulation input-output relations. In *Proceedings of the 1992 Winter Simulation Conference*, 1992.
- [3] R. W. Blanning. The construction and implementation of metamodels. *Simulation*, pages 177–184, june 1975.
- [4] Laura I. Burke and James P. Ignizio. Neural networks and operation research: an overview. *Computer Ops. Res.*, 19(3), 1992.
- [5] G. Chryssolouris, M. Lee, and M. Domroese. The use of neural networks for in determining operational policies for manufacturing systems. *Journal of Manufacturing Systems*, 100(2).
- [6] G. Chryssolouris, M. Lee, J. Pierce, and M. Domroese. Use of neural networks for the design of manufacturing systems. *Manufacturing Review*, 3(3), 1990.
- [7] Judith E. Dayhoff. *Neural network architectures: An introduction*. Van Nostrand Reihold, new york, 1990.
- [8] Paul A. Fishwick. Neural network models in simulation: A comparison with traditional modeling approaches. In *Proceedings of the 1989 Winter Simulation Conference*, 1989.

- [9] L. W. Freidman and I. Pressman. The metamodeling in simulation analysis: can it be trusted? *Journal of Operational Research society*, 39(10), 1988.
- [10] J. Udo Godwin and Yash P. Gupta. Applications of neural networks in manufacturing management systems. *Production Planning and Control*, 5(3), 1994.
- [11] C. M. Harmonosky and S. F. Robohn. Real-time scheduling in computer integrated manufacturing: a review of recent research. *International Journal of Integrated Manufacturing*, 4(6):331–340, 1991.
- [12] C. M. Harmonosky and S. F. Robohn. Investigating the application potential of simulation for real time control decisions. *International Journal of computer Integrated Manufacturing*, 8(2), 1995.
- [13] R. D. Hurrion. Using a neural network to enhance the decision making quality of a visual interactive simulation model. *Journal of Operational Research Society*, 43(4), 1992.
- [14] Y. Kao and Y. B. Moon. A unified group technology implementation using back propagation learning rule of neural networks. *Computers and Industrial Engineering*, 20(4):425–437, 1991.
- [15] S. Kaparthi and N. C. Suresh. A neural network system for shape-based classification and coding of rotational parts. *International Journal of Production Research*, 29(9):1771–1784, 1991.
- [16] Suleyman Karabuk and Ihsan Sabuncuoglu. Issues in integrating simulation and analytical methods in a real time scheduling environment. In Ali R. Kaylan, Axel Lehmann, and Tuncer I. Oren, editors, *Proceedings of European Simulation Symposium, ESS 94*, 1994.
- [17] F. C. John Khaw, B. B. Lim, and L. E. N. Lim. Optimal design of neural networks using the taguchi method. *Neurocomputing*, 7, 1995.

- [18] Robert A. Kilmer, Alice E. Smith, Cenk Tunasr, and Larry J. Shuman. A neural network metamodel of an emergency department simulation. In *4th Industrial Engineering Research Conference Proceedings*.
- [19] George R. Madey, Jay Weinrth, and Vijay Shah. Integration of neuro-computing and system simulation for modeling continuous improvement systems in manufacturing. *Journal of Intelligent Manufacturing*, 3, 1992.
- [20] E. Masson and Y. Wang. Introduction to computation and learning in artificial neural networks. *European Journal of Operational Research*, 47, 1990.
- [21] Y. B. Moon and S. C. Chi. Generalized part family formation using neural network techniques. *Journal of Intelligent Manufacturing*, 3(4):205-216, 1992.
- [22] Dan Murray. Tuning neural networks with generic algorithms. *AI Expert*, 9(6), 1994.
- [23] NeuralWare, Inc. *NeuralWorks Professional II and NeuralWorks Explorer*.
- [24] Mary Lou Padgett and Thaddeus A. Roppel. Neural networks and simulation: Modeling for applications. *Simulation*, pages 295-305, may 1992.
- [25] C. Dennis Pegden, Robert E. Shannon, and Randall P. Sadowski. *Introduction to simulation using SIMAN*. McGRAW-HILL, 1991.
- [26] Patrick R. Philipoon, Loren Paul Rees, and Lars Wiegman. Using neural networks to determine internally set due date assignments for shop scheduling. *Decision Sciences*, 25(5/6):825,851, september 1994.
- [27] Henri Pierreval. A metamodeling approach based on neural networks. *International Journal of Computer Simulation*, special issue.
- [28] Henri Pierreval. Rule based simulation metamodel. *European Journal of Operational Research Society*, 61, 1992.

- [29] Henri Pierreval. Neural networks to select dynamic scheduling heuristics. *Revue des Systemes de Decision*, 2(2), 1993.
- [30] I. Sabuncuoglu, A. Yilmz, and E. Ozkaylar. Input data analysis for simulation using neural networks. In *Proceedings of the Advances in Simulation '92 Symposium*, 1992.
- [31] Ihsan Sabuncuoglu and Don Hommertzheimer. Expert systems and simulation in flexible manufacturing systems. *Computers Industrial Engineering*, 15:1-7, 1988.
- [32] Ihsan Sabuncuoglu and Don L. Hommertzheimer. Experimental investigation of an fms due date scheduling problem: Evaluation of machine and agv scheduling rules. *The International Journal of Flexible Manufacturing Systems*, 5:301-323, 1993.
- [33] Robert G. Sargent. Research issues in metamodeling. In *Proceedings of the 1991 Winter Simulation Conference*, 1991.
- [34] Patrick J. Starr. Integration of simulation and analytical submodels for supporting manufacturing decisions. *International Journal of Production Research*, 29(9):1733-1746, 1991.
- [35] B. Yu and K. Poplewell. Metamodeling in manufacturing: a review. *International Journal of Production Research*, 32(4), 1994.
- [36] Fatemah Zehidi. An introduction to neural networks and a comparison with artificial intelligence and expert systems. *Interfaces*, 23, 1991.
- [37] H. C. Zhang and S. H. Huang. Application of neural networks in manufacturing: a state of the art survey. *International Journal of Production Research*, 33(3), 1995.

Appendix A

Figures

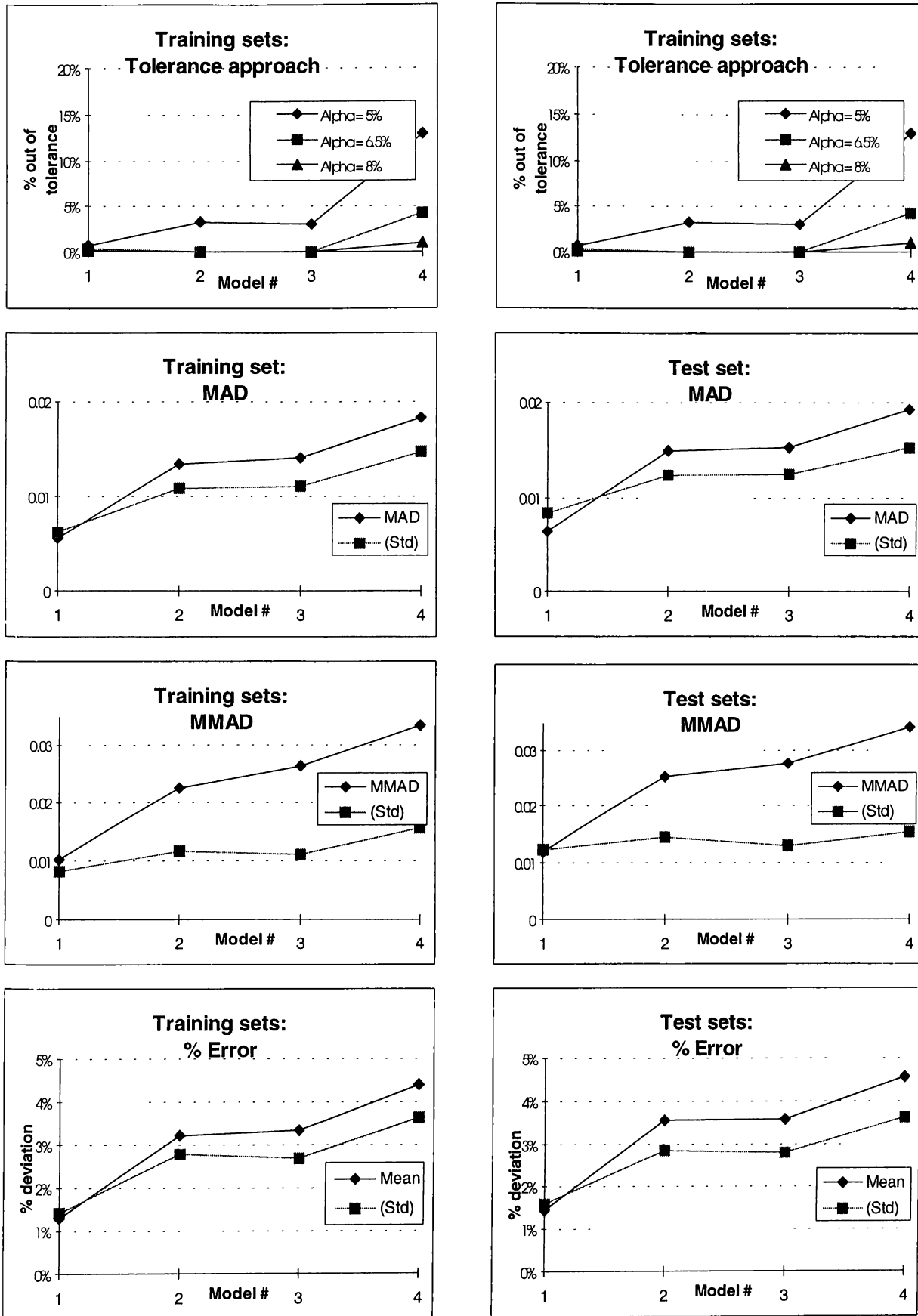
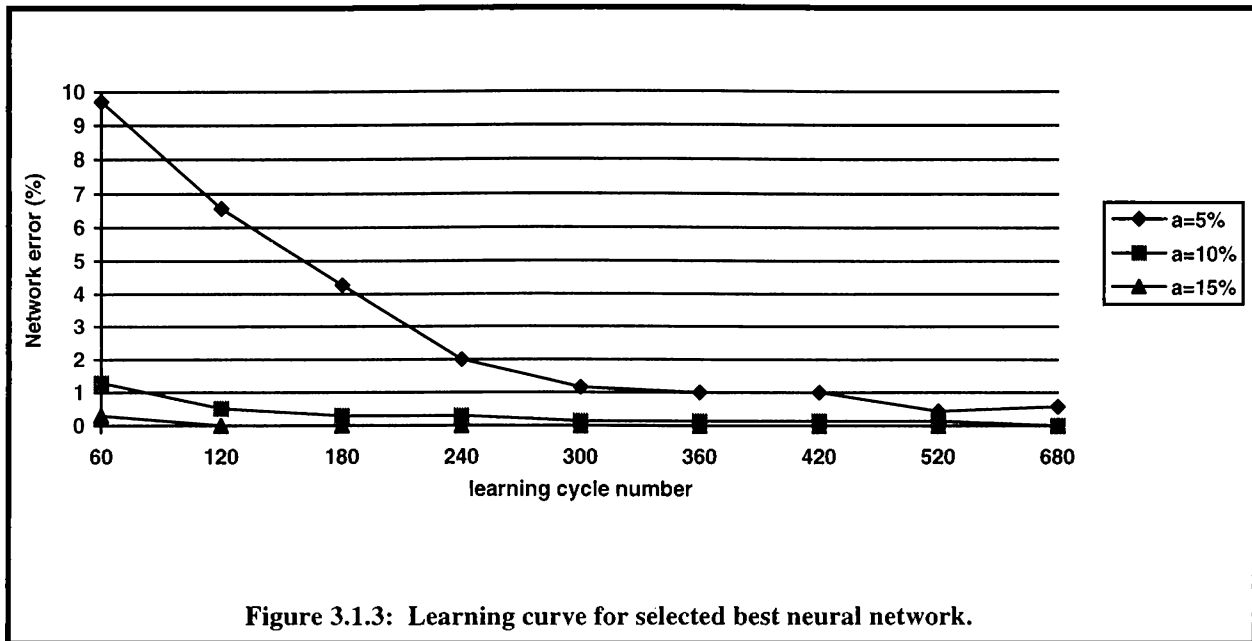


Figure 3.1.2: Mean Utilization: Simple System: Metamodel performance through assesement criteria.



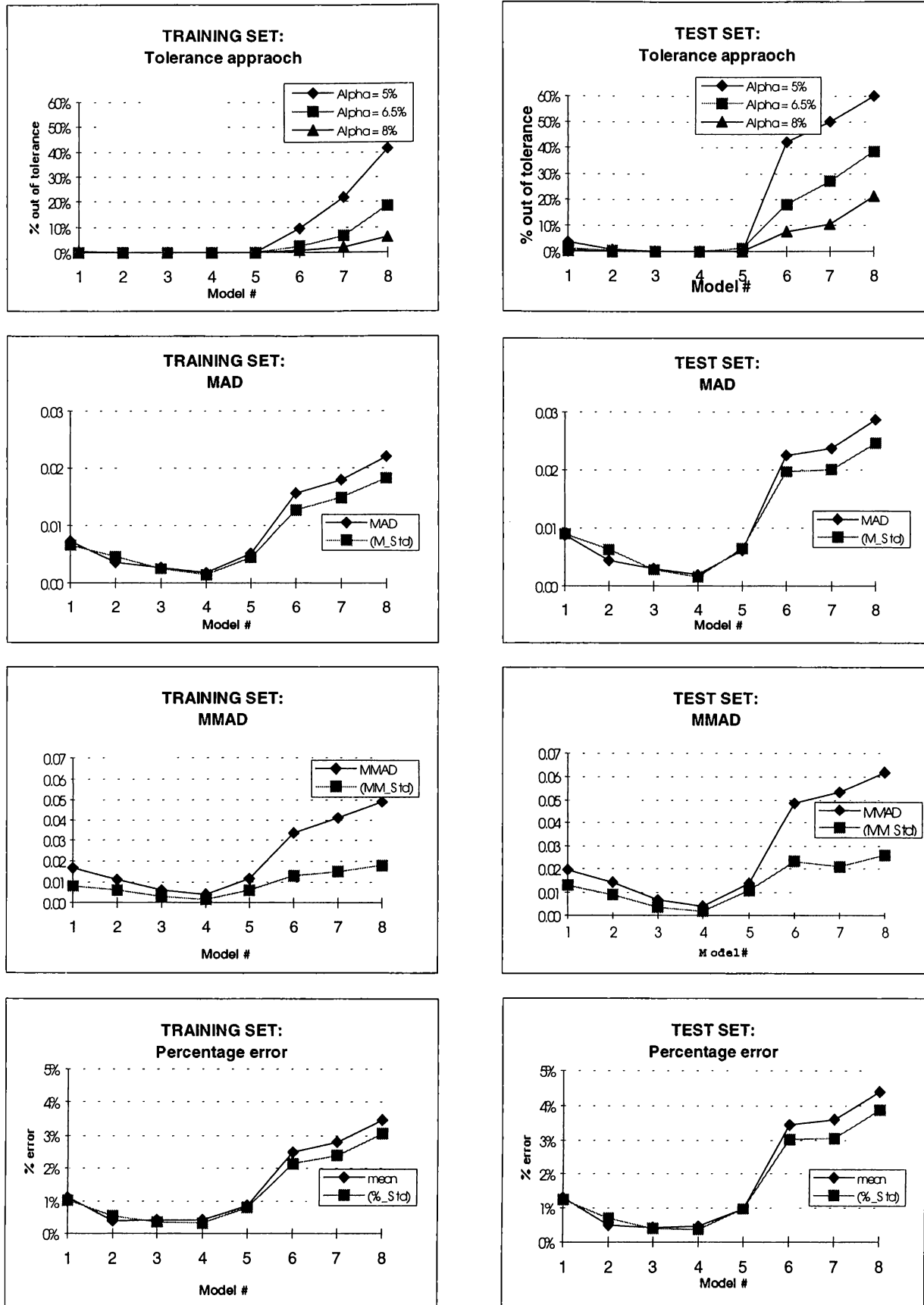


Figure 3.2.2: Mean Utilization: Complex System: Metamodel performance through assesment criteria.

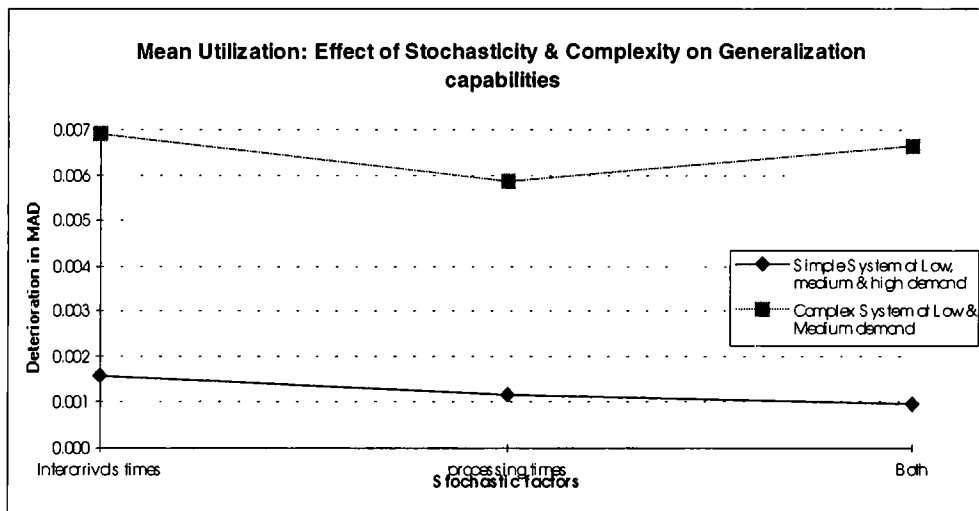
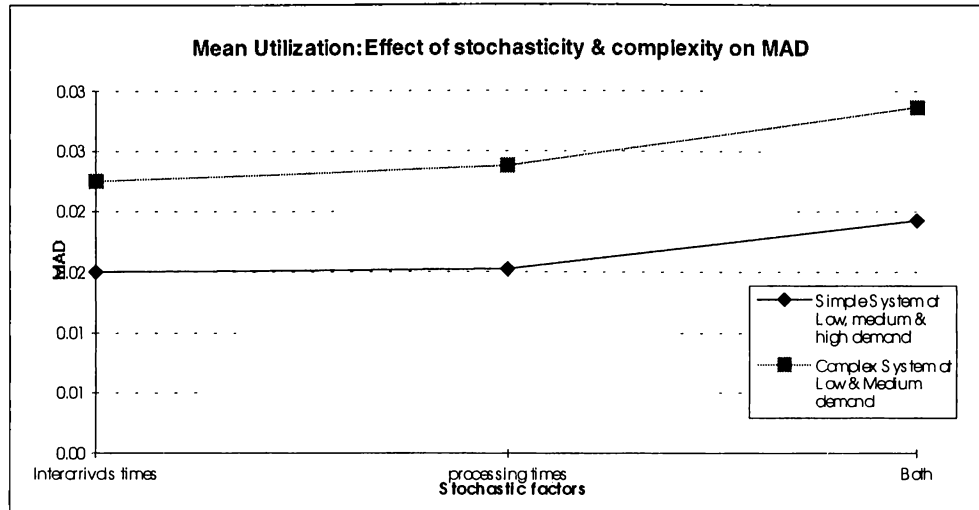


Figure 3.3: Mean Utilization: Comparisons.

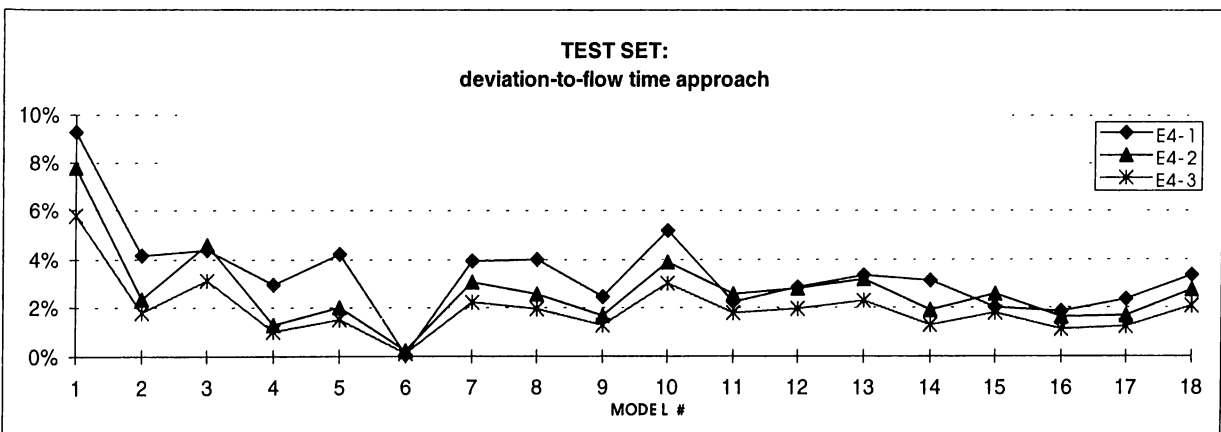
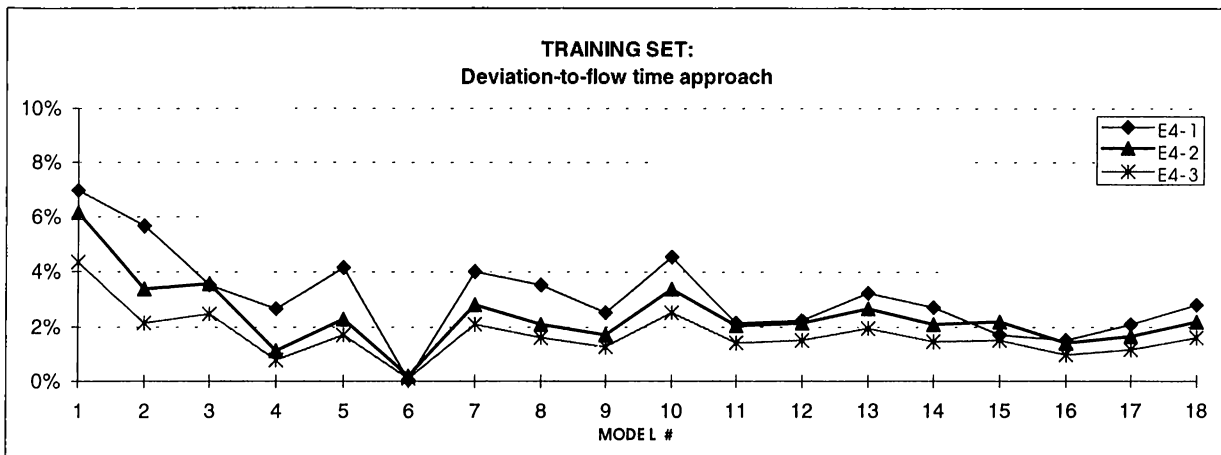
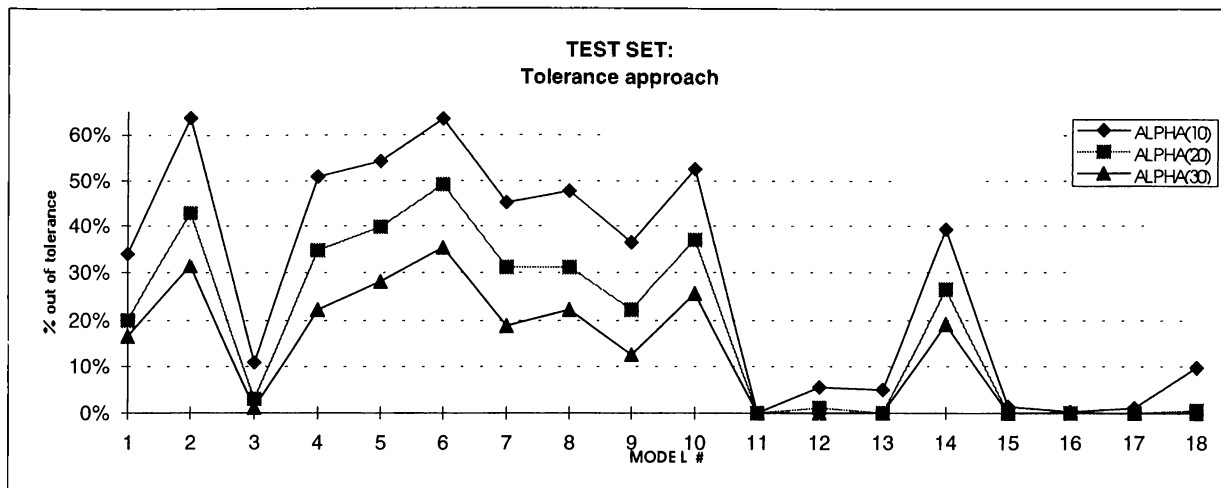
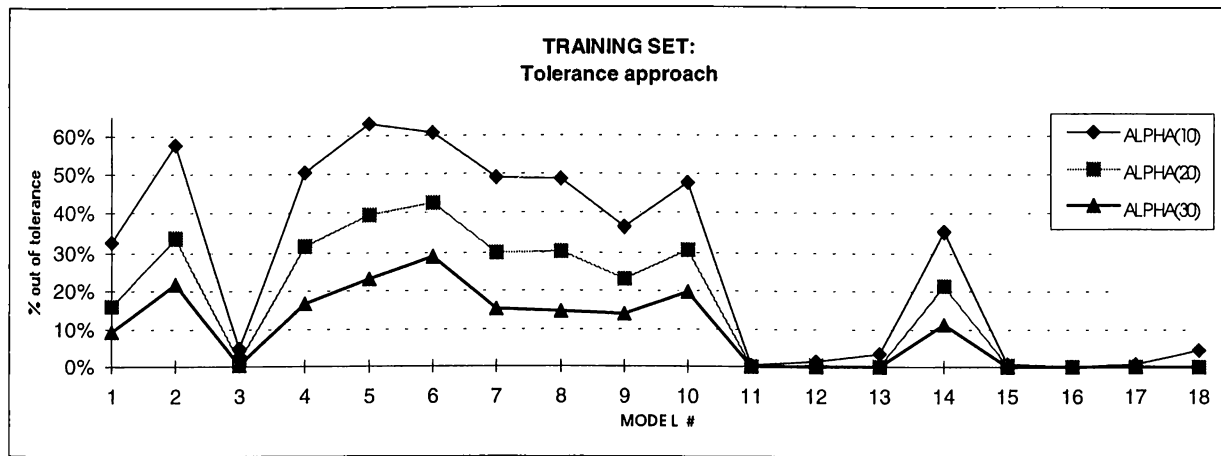


Figure 4.1.2: Mean Tardiness: simple System: Metamodel performance through assesment criteria.

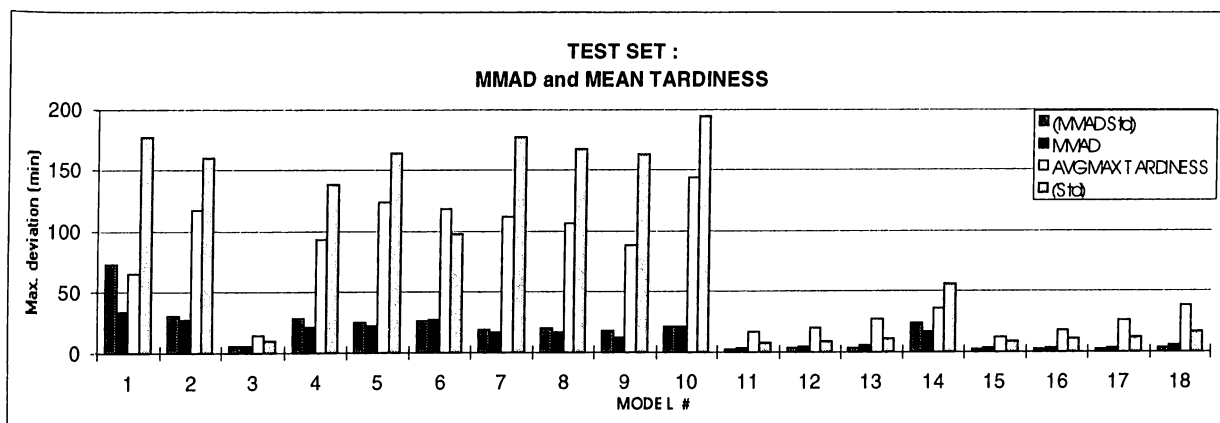
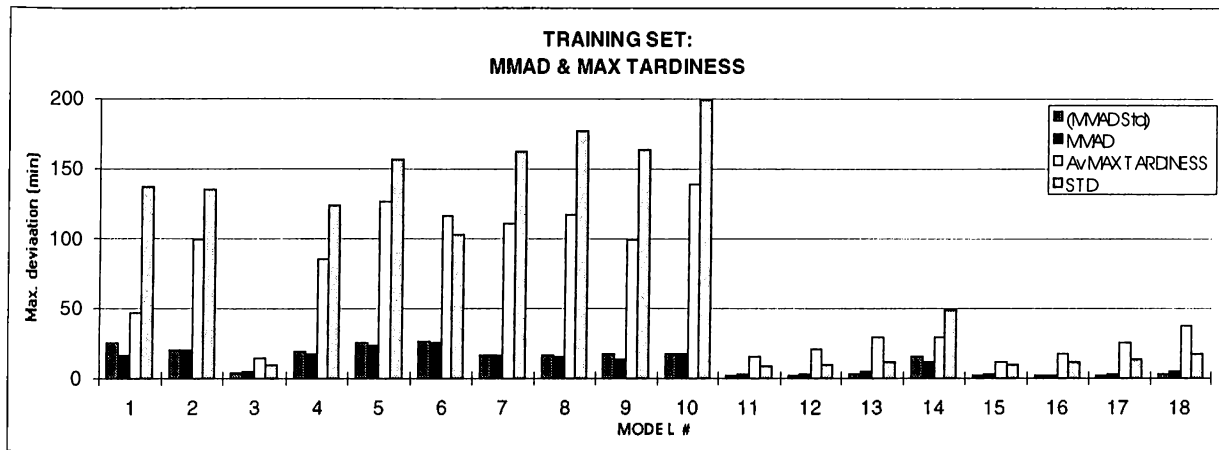
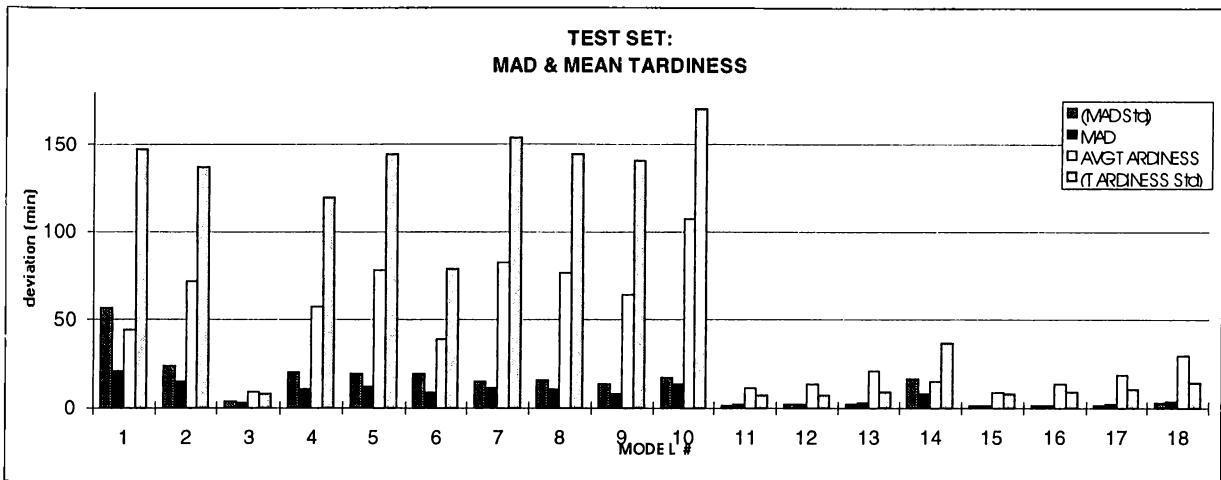
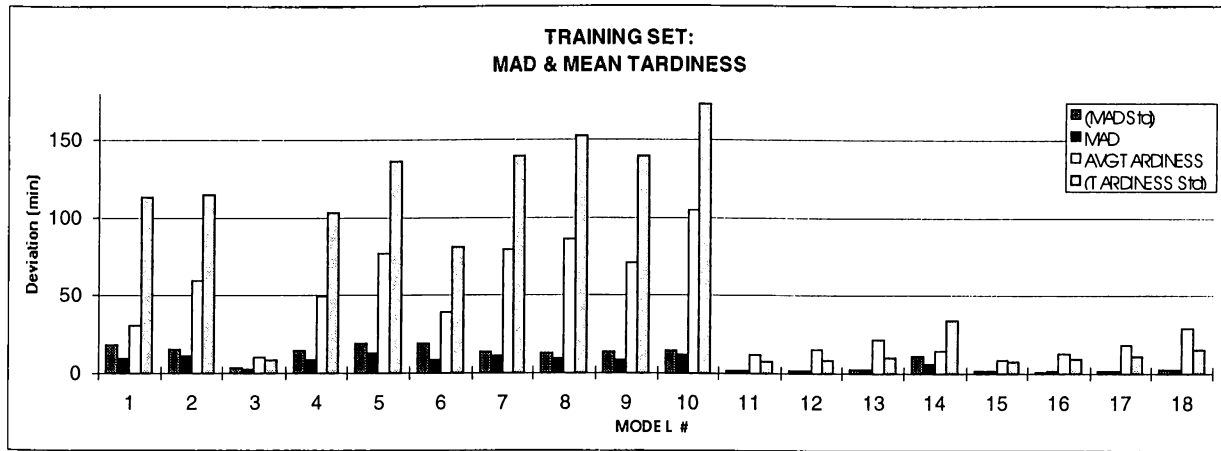


Figure 4.1.2 (Cont'd): Mean Tardiness: simple Syste: Metamodel performance through assesment criteria.

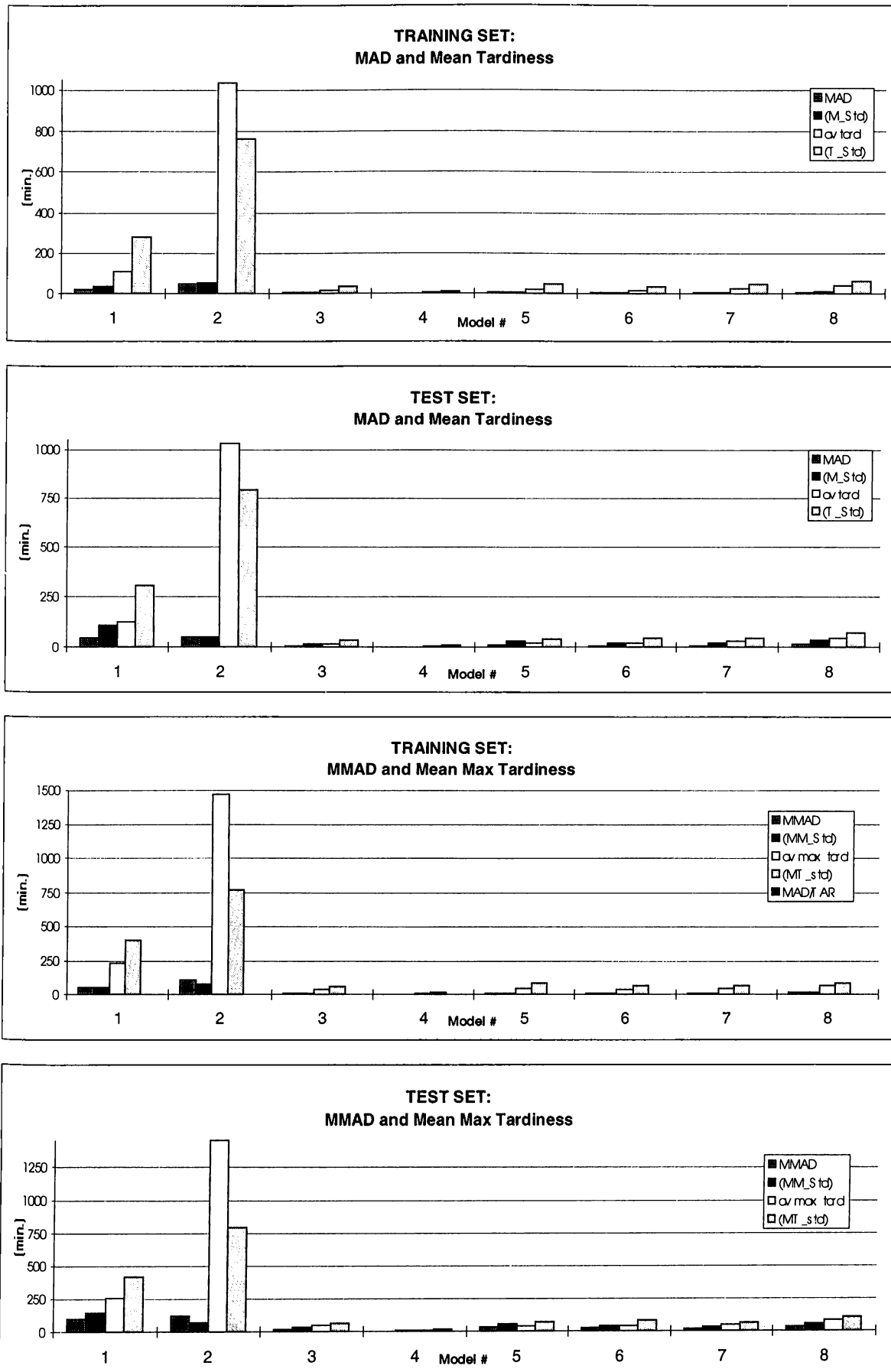


Figure 4.2.1: Mean Tardiness: Complex System: Metamodel performance through assesment criteria.

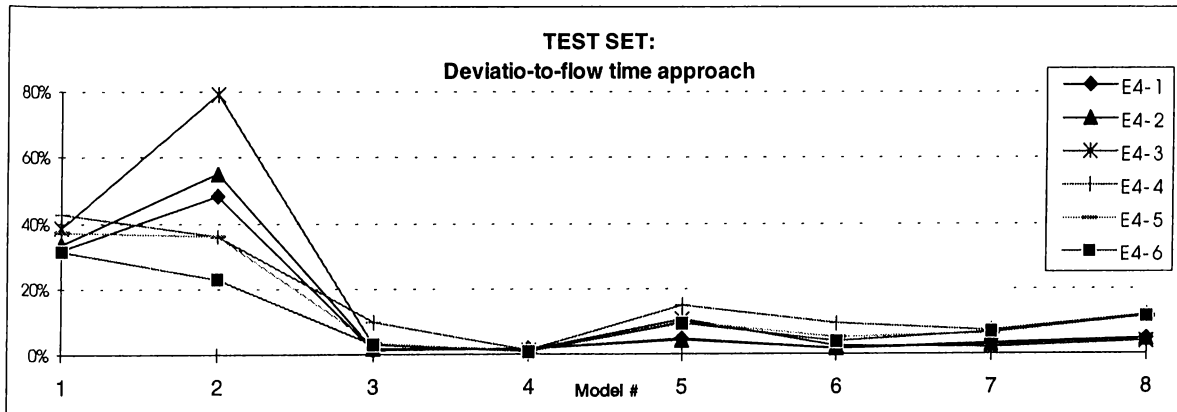
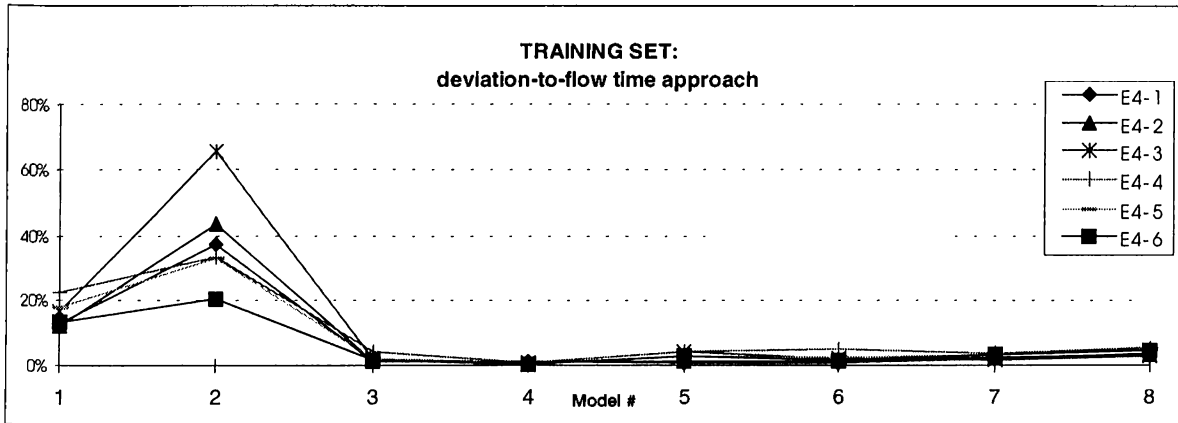
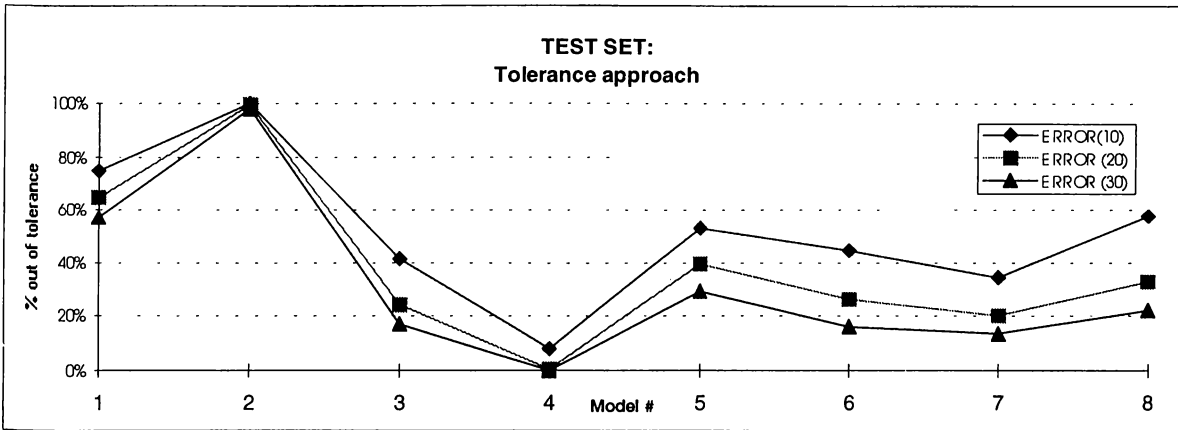
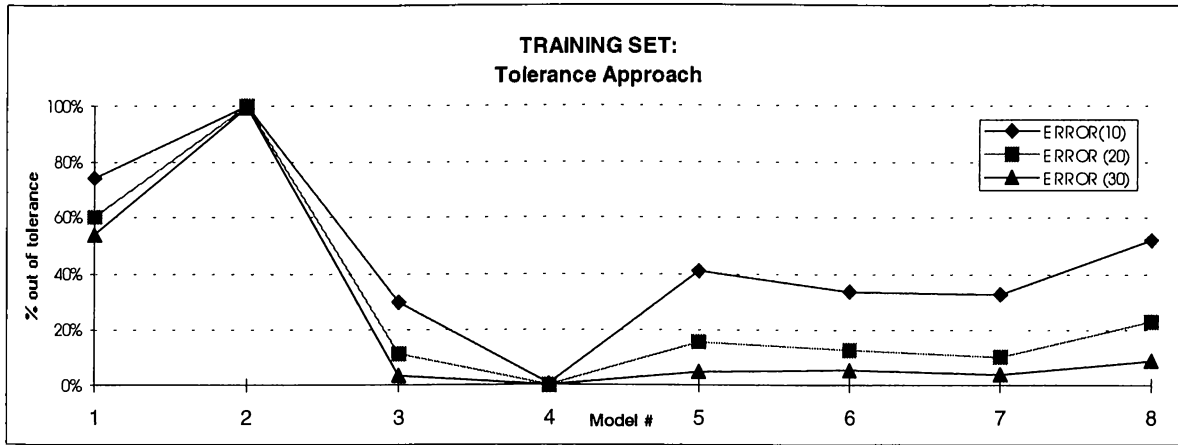


Figure 4.2.1 (Cont'd): Mean Tardiness: Complex System: Metamodel performance through assesment criteria.

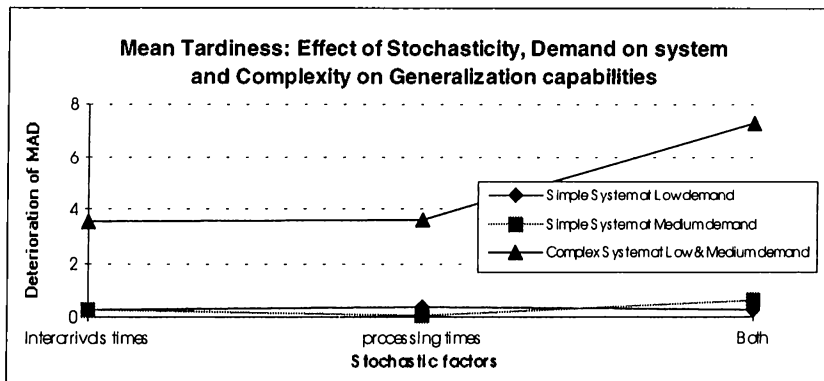
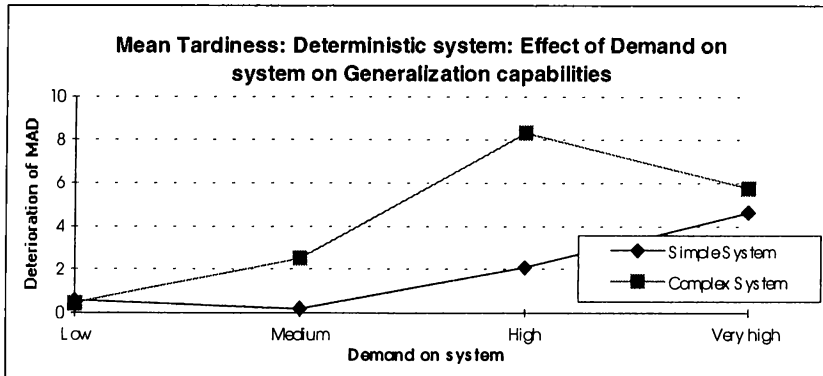
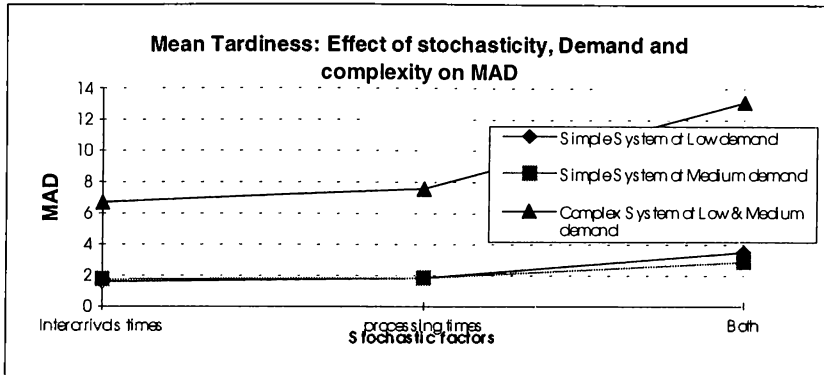
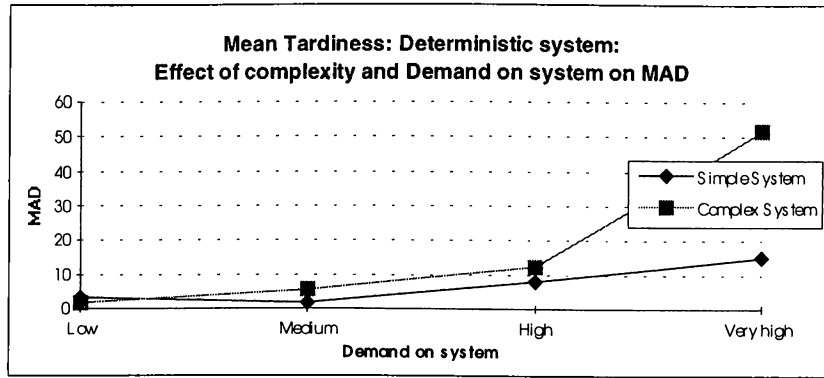


Figure 4.3: Mean Tardiness: Comparisons.

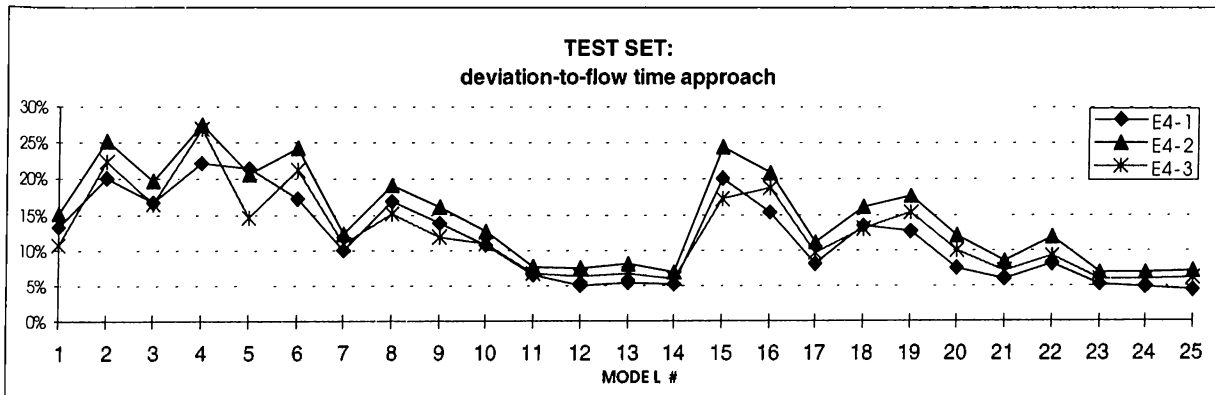
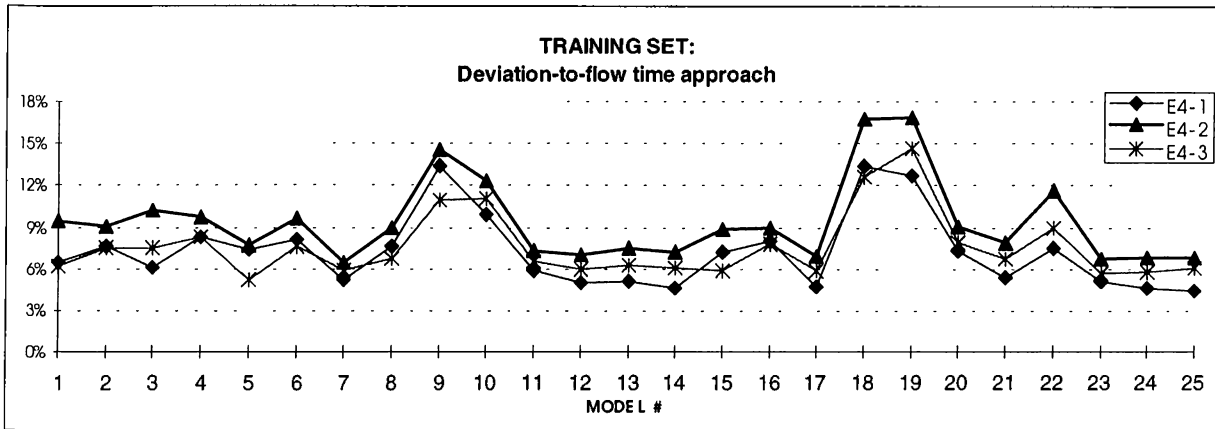
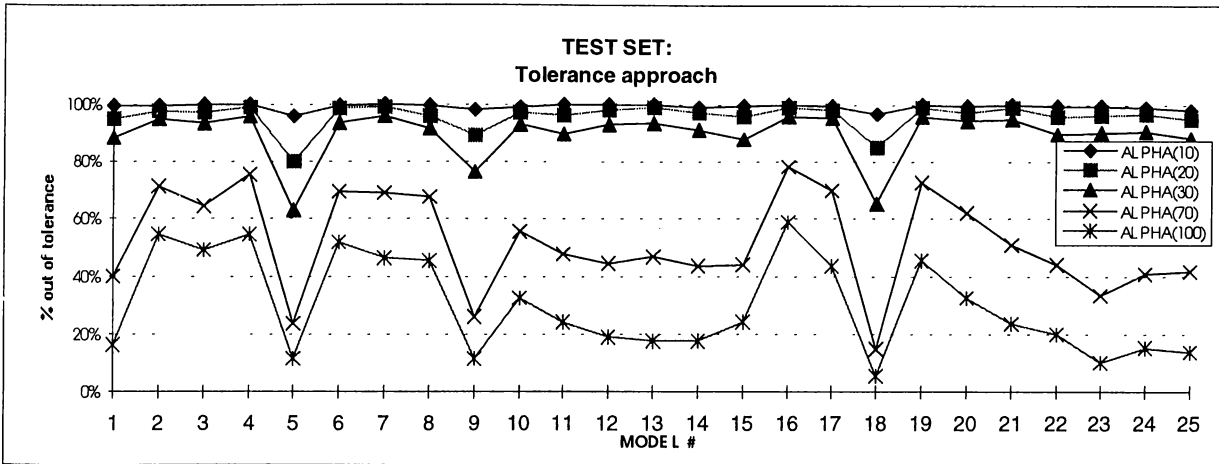
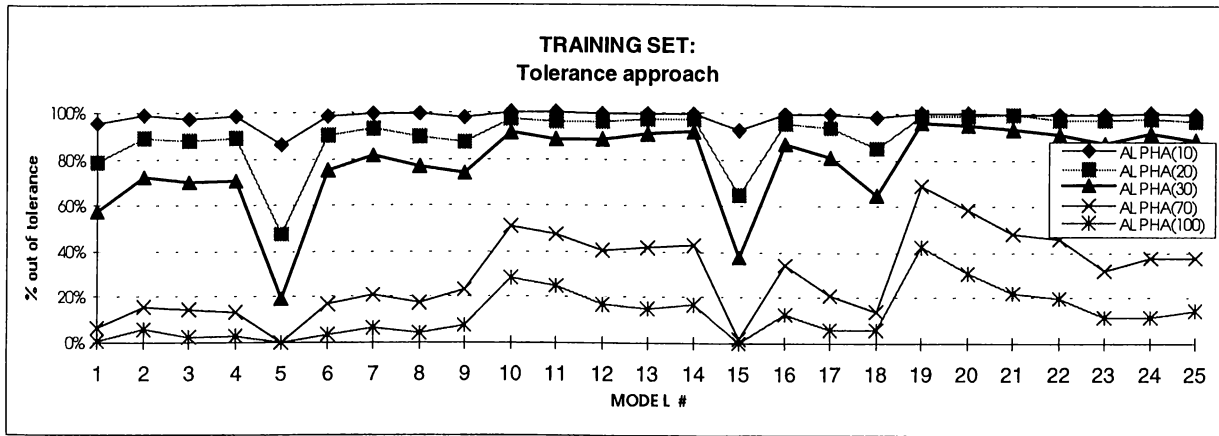


Figure 5.2: Mean Tardiness: Simple System: Short term performance estimation: Metamodel performance through assesment criteria.

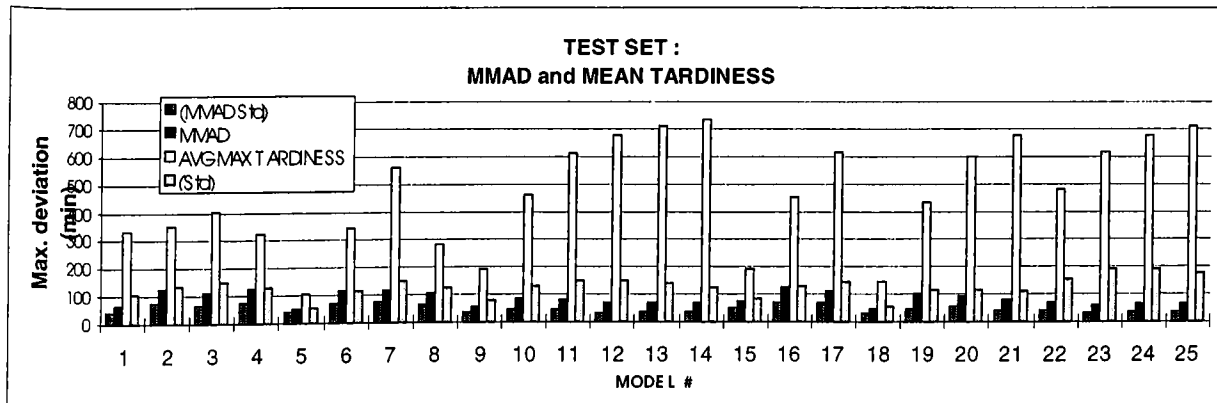
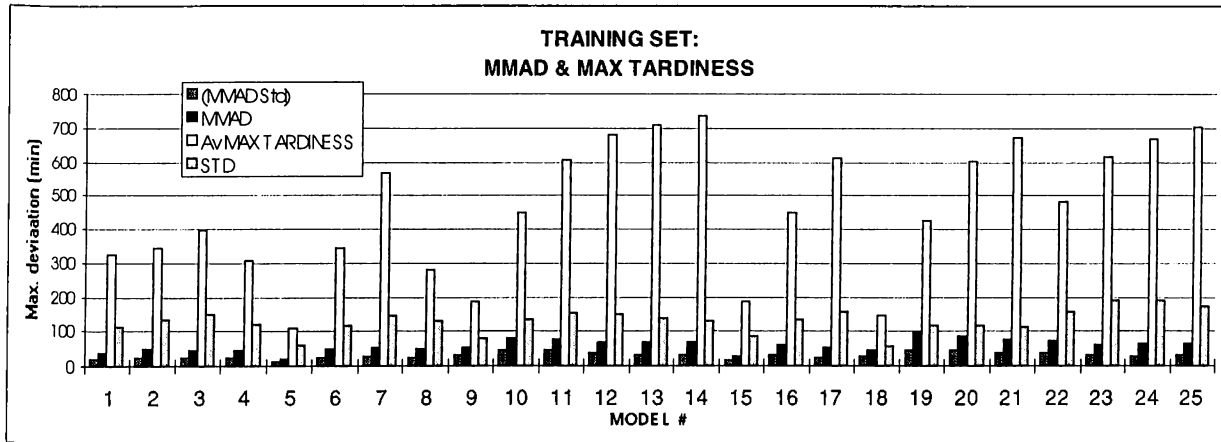
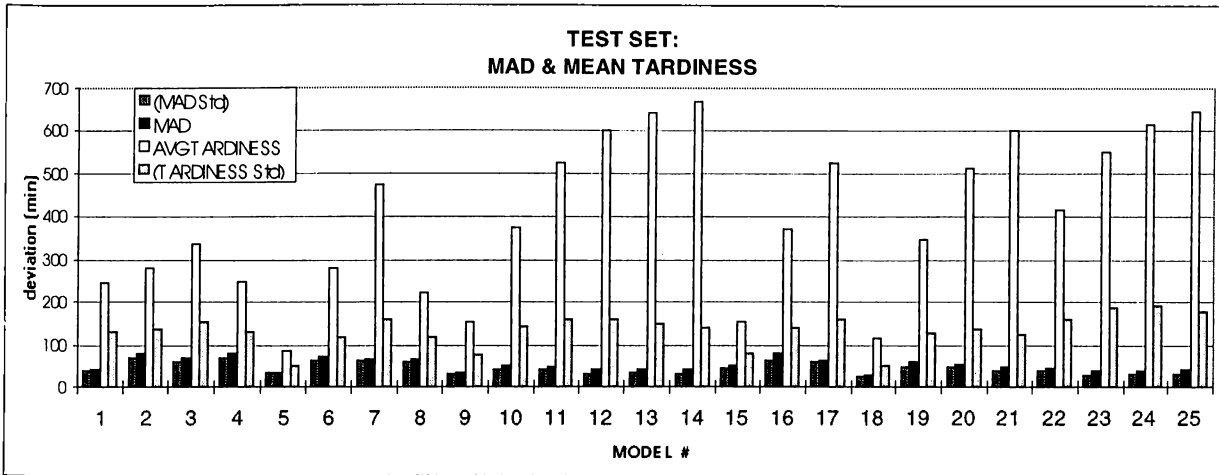
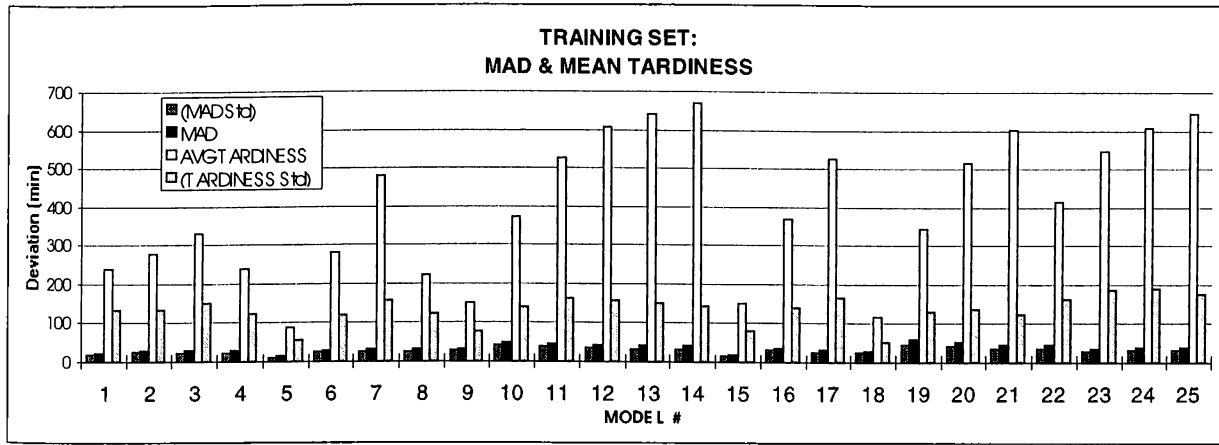


Figure 5.2 (Cont'd): Mean Tardiness: Simple System: Short term performance estimation: Metamodel performance through assesement criteria.

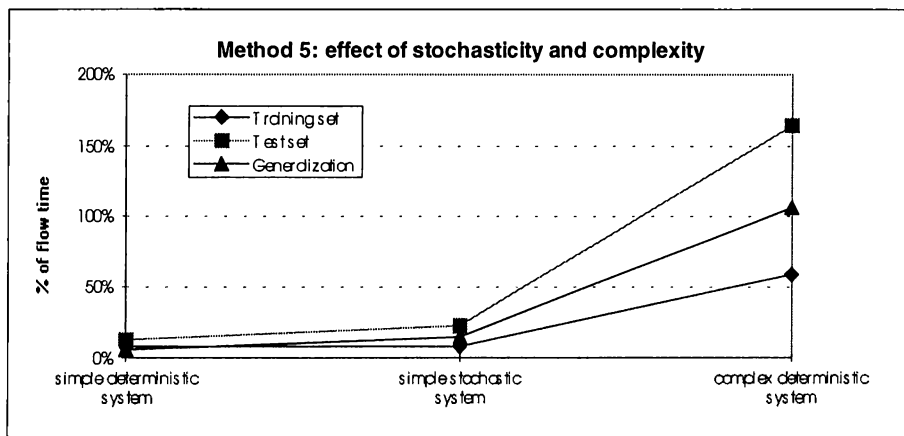
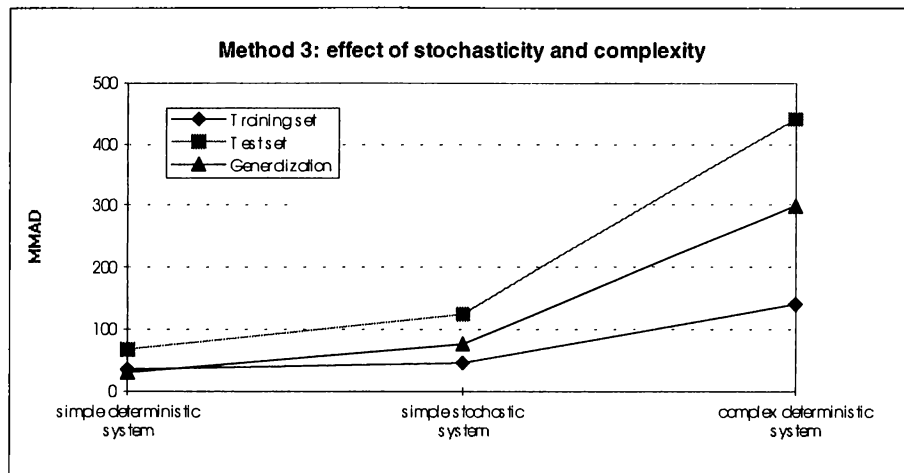
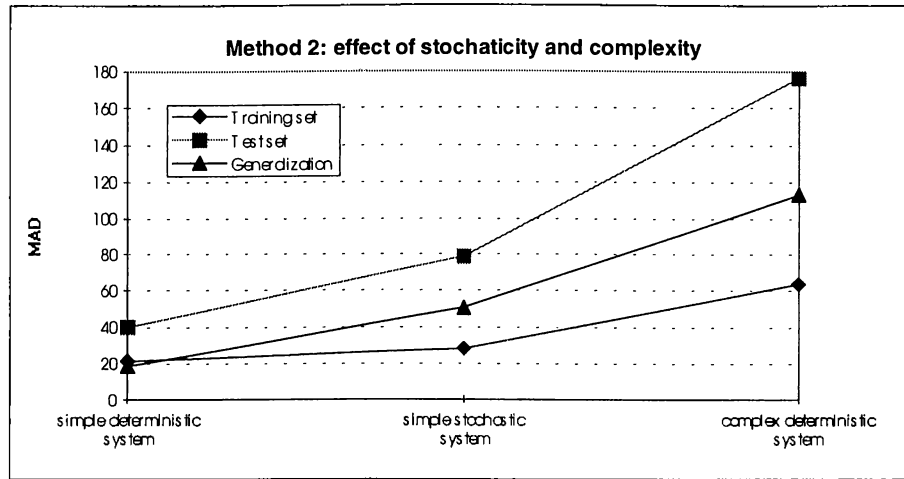


Figure 5.3: Mean Tardiness: Short term performance estimation: Comparing effect of stochasticity and system complexity.

Appendix B

Tables

MODEL #	Training set #	Test set #	Interarrival time		Processing time Nature	Scheduling rule
			Nature	Range		
1	1	1, 5 and 6	deterministic	[14..85]	deterministic	SPT or FCFS
2	2	2	stochastic (exponential)	mean in [14..85]	deterministic	SPT or FCFS
3	3	3	deterministic	[60..85]	stochastic (exponential)	SPT or FCFS
4	4	4	stochastic (exponential)	mean in [14..85]	stochastic (exponential)	SPT or FCFS

Table 3.1.1: Mean Utilization: Simple System: List of models.

Model #	#	NNet	Size	Training set #	(size)	Test set #	(size)	learning rate	momentum rate	examples trained on
Model 1	1	expl_1_1	4_6_7_6_4	#1	700	#1	400	0.9	0.6	280000
	2	expl_1_2	4_6_7_6_4	#1	700	#1	400	0.9	0.6	500000
	3	expl_1_3	4_6_7_6_4	#1	700	#1	400	0.9	0.6	280000
	4	expl_1_4	4_6_7_6_4	#1	700	#1	400	0.9	0	280000
	5	expl_1_5	4_6_7_6_4	#1	700	#1	400	0.9	0	500000
	6	expl_1_6	4_6_7_6_4	#1	700	#1	400	1.5	0	280000
	7	expl_1_7	4_6_7_6_4	#1	700	#1	400	1.5	0	500000
	8	expl_1_8	4_6_7_6_4	#1	700	#1	400	0.3	0	500000
	9	expl_1_9	4_6_7_6_4	#1	700	#1	400	1.2	0.6	500000
	10	expl_1_11	4_6_7_6_4	#1	700	#1	400	0.3	0.3	500000
Model 2	11	expl_2_1	4_6_7_6_4	#2	400	#2	200	0.9	0.6	500000
	12	expl_2_2	4_55_4	#2	400	#2	200	0.9	0.6	500000
	13	expl_2_3	4_8_9_8_4	#2	400	#2	200	0.9	0.6	500000
Model 3	14	expl_3_1	4_6_7_6_4	#3	400	#3	200	0.9	0.6	500000
	15	expl_3_2	4_55_4	#3	400	#3	200	0.9	0.6	500000
	16	expl_3_3	4_8_9_8_4	#3	400	#3	200	0.9	0.6	500000
Model 4	17	expl_4_1	4_6_7_6_4	#4	400	#4	200	0.9	0.6	500000
	18	expl_4_2	4_55_4	#4	400	#4	200	0.9	0.6	500000
	19	expl_4_3	4_8_9_8_4	#4	400	#4	200	0.9	0.6	500000

Table 3.1.2: Mean Utilization: Simple System: List of neural metamodels.

#	NNet	Data set	size	METHOD 1:			TOLERANCE approach			METHOD 2:	
				Alpha = 5%	Alpha = 6.5%	Alpha = 8%	min	MAD	(Std)	max	
1	exp1_1_1	Training set	700	1.1%	0.4%	0.1%	0.000	0.007	0.007	0.092	
		Test set	400	2.0%	1.0%	0.5%	0.000	0.008	0.010	0.149	
2	exp1_1_2	Training set	700	0.7%	0.3%	0.1%	0.000	0.006	0.006	0.084	
		Test set	400	1.8%	1.0%	0.5%	0.000	0.006	0.008	0.133	
3	exp1_1_3	Training set	700	2.3%	1.0%	0.6%	0.000	0.010	0.009	0.128	
		Test set	400	4.3%	1.8%	0.5%	0.000	0.011	0.010	0.107	
4	exp1_1_4	Training set	700	5.0%	2.1%	0.9%	0.000	0.014	0.011	0.140	
		Test set	400	9.3%	4.3%	1.8%	0.000	0.015	0.013	0.106	
5	exp1_1_5	Training set	700	2.4%	0.9%	0.7%	0.000	0.010	0.010	0.161	
		Test set	400	5.3%	2.3%	0.5%	0.000	0.011	0.012	0.105	
6	exp1_1_6	Training set	700	3.1%	1.1%	0.7%	0.000	0.011	0.011	0.162	
		Test set	400	6.5%	2.0%	0.3%	0.000	0.012	0.013	0.083	
7	exp1_1_7	Training set	700	2.1%	0.7%	0.6%	0.000	0.012	0.010	0.153	
		Test set	400	4.5%	1.5%	0.8%	0.000	0.013	0.011	0.091	
8	exp1_1_8	Training set	700	6.9%	2.4%	1.6%	0.000	0.015	0.013	0.142	
		Test set	400	9.5%	5.8%	3.3%	0.000	0.016	0.015	0.121	
9	exp1_1_9	Training set	700	1.3%	0.7%	0.1%	0.000	0.008	0.008	0.102	
		Test set	400	1.8%	0.8%	0.5%	0.000	0.009	0.009	0.106	
10	exp1_1_10	Training set	700	2.7%	1.3%	0.7%	0.000	0.010	0.010	0.130	
		Test set	400	6.3%	2.3%	0.8%	0.000	0.011	0.011	0.103	
11	exp1_2_1	Training set	400	7.5%	2.8%	0.3%	0.000	0.018	0.014	0.085	
		Test set	200	9.0%	3.0%	1.0%	0.000	0.018	0.015	0.128	
12	exp1_2_2	Training set	400	3.3%	0.0%	0.0%	0.000	0.013	0.011	0.064	
		Test set	200	2.5%	0.5%	0.5%	0.000	0.015	0.012	0.147	
13	exp1_2_3	Training set	400	5.5%	1.0%	0.3%	0.000	0.017	0.013	0.091	
		Test set	200	10.5%	3.5%	1.0%	0.000	0.018	0.015	0.108	
14	exp1_3_1	Training set	400	6.3%	1.3%	0.3%	0.000	0.015	0.012	0.097	
		Test set	200	10.5%	4.0%	1.0%	0.000	0.016	0.014	0.100	
15	exp1_3_2	Training set	400	7.8%	2.0%	0.3%	0.000	0.017	0.013	0.085	
		Test set	200	12.0%	3.0%	0.5%	0.000	0.018	0.014	0.086	
16	exp1_3_3	Training set	400	3.0%	0.0%	0.0%	0.000	0.014	0.011	0.064	
		Test set	200	5.0%	2.0%	0.5%	0.000	0.015	0.012	0.096	
17	exp1_4_1	Training set	400	14.8%	4.3%	1.0%	0.000	0.018	0.015	0.099	
		Test set	200	21.0%	3.5%	1.5%	0.000	0.020	0.016	0.086	
18	exp1_4_2	Training set	400	13.0%	4.3%	1.0%	0.000	0.018	0.015	0.096	
		Test set	200	18.0%	4.5%	0.0%	0.000	0.019	0.015	0.076	
19	exp1_4_3	Training set	400	12.0%	2.5%	1.0%	0.000	0.017	0.014	0.094	
		Test set	200	15.5%	4.5%	1.5%	0.000	0.019	0.016	0.140	

Table 3.1.3: Mean Utilization: Simple System: Results of metamodels.

#	NNet	Data set	METHOD 3:				METHOD 4:			% Error	
			min	MMAD	(Std)	max	min	Average	(Std)	max	
1	exp1_1_1	Training set	0.001	0.013	0.0096	0.092	0.0%	1.7%	1.7%	18.1%	
		Test set	0.002	0.014	0.0135	0.149	0.0%	1.8%	1.9%	18.5%	
2	exp1_1_2	Training set	0.001	0.010	0.0083	0.084	0.0%	1.3%	1.4%	14.0%	
		Test set	0.002	0.012	0.0123	0.133	0.0%	1.4%	1.6%	13.3%	
3	exp1_1_3	Training set	0.003	0.019	0.0115	0.128	0.0%	2.5%	2.4%	25.1%	
		Test set	0.002	0.020	0.0131	0.107	0.0%	2.7%	2.5%	21.2%	
4	exp1_1_4	Training set	0.002	0.024	0.0139	0.140	0.0%	3.5%	3.1%	26.2%	
		Test set	0.003	0.026	0.0158	0.106	0.0%	3.6%	3.2%	23.3%	
5	exp1_1_5	Training set	0.001	0.018	0.0139	0.161	0.0%	2.3%	2.7%	22.7%	
		Test set	0.003	0.020	0.0154	0.105	0.0%	2.6%	2.8%	19.7%	
6	exp1_1_6	Training set	0.002	0.021	0.0151	0.162	0.0%	2.6%	2.7%	23.0%	
		Test set	0.003	0.023	0.0152	0.083	0.0%	2.9%	2.9%	18.7%	
7	exp1_1_7	Training set	0.004	0.020	0.0122	0.153	0.0%	3.0%	2.4%	21.6%	
		Test set	0.004	0.022	0.0129	0.091	0.0%	3.2%	2.5%	15.9%	
8	exp1_1_8	Training set	0.004	0.026	0.0163	0.142	0.0%	3.7%	3.7%	27.7%	
		Test set	0.003	0.028	0.0186	0.121	0.0%	3.8%	3.8%	24.8%	
9	exp1_1_9	Training set	0.001	0.015	0.0104	0.102	0.0%	2.1%	2.2%	19.4%	
		Test set	0.002	0.016	0.0120	0.106	0.0%	2.2%	2.4%	20.3%	
10	exp1_1_10	Training set	0.002	0.019	0.0132	0.130	0.0%	2.5%	2.7%	21.0%	
		Test set	0.002	0.020	0.0147	0.103	0.0%	2.6%	2.8%	18.4%	
11	exp1_2_1	Training set	0.003	0.028	0.0144	0.085	0.0%	4.2%	3.1%	21.1%	
		Test set	0.005	0.029	0.0160	0.128	0.0%	4.3%	3.3%	34.8%	
12	exp1_2_2	Training set	0.004	0.022	0.0117	0.064	0.0%	3.2%	2.8%	19.7%	
		Test set	0.004	0.025	0.0144	0.147	0.0%	3.5%	2.8%	24.4%	
13	exp1_2_3	Training set	0.002	0.027	0.0128	0.091	0.0%	4.1%	3.0%	16.0%	
		Test set	0.004	0.029	0.0163	0.108	0.0%	4.1%	3.1%	26.2%	
14	exp1_3_1	Training set	0.005	0.028	0.0127	0.097	0.0%	3.6%	3.2%	28.7%	
		Test set	0.008	0.030	0.0150	0.100	0.0%	3.9%	3.6%	27.8%	
15	exp1_3_2	Training set	0.006	0.030	0.0130	0.085	0.0%	3.9%	2.9%	18.0%	
		Test set	0.007	0.032	0.0145	0.086	0.0%	4.1%	3.1%	16.9%	
16	exp1_3_3	Training set	0.004	0.026	0.0111	0.064	0.0%	3.4%	2.7%	17.5%	
		Test set	0.007	0.028	0.0130	0.096	0.0%	3.6%	2.8%	16.8%	
17	exp1_4_1	Training set	0.009	0.034	0.0155	0.099	0.0%	4.3%	3.6%	23.2%	
		Test set	0.008	0.035	0.0160	0.086	0.0%	4.7%	3.6%	22.1%	
18	exp1_4_2	Training set	0.004	0.033	0.0155	0.096	0.0%	4.4%	3.6%	23.3%	
		Test set	0.004	0.034	0.0153	0.076	0.0%	4.6%	3.6%	25.8%	
19	exp1_4_3	Training set	0.003	0.032	0.0148	0.094	0.0%	4.3%	3.6%	25.0%	
		Test set	0.006	0.035	0.0177	0.140	0.0%	4.6%	3.8%	32.9%	

Table 3.1.3 (cont'd): Mean Utilization: Simple System: Results of metamodels.

	Error at $\alpha=5\%$	Error at $\alpha=10\%$
Training set (set1)	3%	0%
Test set (set2)	3%	0%

Table 3.1.4: Error report of Pierreval's network

	Error at $\alpha=5\%$	Error at $\alpha=10\%$	MAD	(Std)	MMAD	(Std)	Mean % error	(Std)
SET # 5	68%	0%	0.04	0.012	0.051	0.005	3.8%	5.5%
SET # 6	100%	0%	0.06	0.016	0.081	0.005	5.3%	10.4%

Table 3.1.5: Robustness test

MOD #	Training set	Test set	# of job types	# of mach	Interarrival time		Processing time range	Due date tightness factor	Scheduling rule
					Nature	Range			
1	#1	#1	6	7	deterministic	[20..100]	deterministic	[2..9]	SPT, MODD or EDD
2	#2	#2	6	7	deterministic	[20..40]	deterministic	[2..9]	SPT, MODD or EDD
3	#3	#3	6	7	deterministic	[40..70]	deterministic	[2..9]	SPT, MODD or EDD
4	#4	#4	6	7	deterministic	[70..100]	deterministic	[2..9]	SPT, MODD or EDD
5	#5	#5	6	7	deterministic	[20..100]	deterministic	[2..9]	SPT, MODD or EDD
					*machine utilization < 98%				
6	#6	#6	6	7	stochastic (exponential)	mean in [20..100]	deterministic	[2..9]	SPT, MODD or EDD
7	#7	#7	6	7	deterministic	[20..100]	stochastic (exponential)	[2..9]	SPT, MODD or EDD
8	#8	#8	6	7	stochastic (exponential)	mean in [20..100]	stochastic (exponential)	[2..9]	SPT, MODD or EDD

Table 3.2.1: Mean Utilization: Complex System: List of models.

MODEL #	Percentage of tardy jobs				Machine utilization			
	min.	mean	(std)	max.	min.	mean	(std)	max.
1	0.0%	34.1%	37.1%	100.0%	30.0%	68.5%	17.5%	100.0%
2	0.0%	83.3%	29.7%	100.0%	69.5%	94.7%	6.8%	100.0%
3	0.0%	22.1%	30.0%	100.0%	41.5%	65.8%	11.0%	100.0%
4	0.0%	12.9%	19.9%	90.1%	30.0%	41.8%	7.3%	60.0%
5	0.0%	22.4%	29.9%	100.0%	29.6%	62.5%	14.6%	97.9%
6	0.0%	41.6%	37.2%	100.0%	30.2%	68.1%	17.2%	100.0%
7	0.6%	52.7%	33.8%	100.0%	30.3%	68.1%	17.2%	100.0%
8	0.6%	58.3%	32.0%	100.0%	29.9%	68.1%	17.3%	100.0%

Table 3.2.2: Mean Utilization: Complex System: Generated set characteristics.

	NNet	Size	Training set #	(size)	Test set #	(size)	learning coef	momentum coef	examples learned	
Model 1	1	e2_1_1	8_12_14_12_7	#1	600	#1	400	0.9	0.6	500000
	2	e2_1_2	8_15_19_15_7	#1	600	#1	400	0.9	0.6	500000
	3	e2_1_3	8_45_7	#1	600	#1	400	0.9	0.6	500000
Model 2	4	e2_2_1	8_12_14_12_7	#2	600	#2	400	0.9	0.6	500000
	5	e2_2_2	8_15_19_15_7	#2	600	#2	400	0.9	0.6	500000
	6	e2_2_3	8_45_7	#2	600	#2	400	0.9	0.6	500000
Model 3	7	e2_3_1	8_12_14_12_7	#3	600	#3	400	0.9	0.6	500000
	8	e2_3_2	8_15_19_15_7	#3	600	#3	400	0.9	0.6	500000
	9	e2_3_3	8_45_7	#3	600	#3	400	0.9	0.6	500000
Model 4	10	e2_4_1	8_12_14_12_7	#4	600	#4	400	0.9	0.6	500000
	11	e2_4_2	8_15_19_15_7	#4	600	#4	400	0.9	0.6	500000
	12	e2_4_3	8_45_7	#4	600	#4	400	0.9	0.6	500000
Model 5	13	e2_5_1	8_12_14_12_7	#5	400	#5	240	0.9	0.6	500000
	14	e2_5_2	8_15_19_15_7	#5	400	#5	240	0.9	0.6	500000
	15	e2_5_3	8_45_7	#5	400	#5	240	0.9	0.6	500000
Model 6	16	e2_6_1	8_12_14_12_7	#6	400	#6	200	0.9	0.6	500000
	17	e2_6_2	8_15_19_15_7	#6	400	#6	200	0.9	0.6	500000
	18	e2_6_3	8_45_7	#6	400	#6	200	0.9	0.6	500000
Model 7	19	e2_7_1	8_12_14_12_7	#7	400	#7	200	0.9	0.6	500000
	20	e2_7_2	8_15_19_15_7	#7	400	#7	200	0.9	0.6	500000
	21	e2_7_3	8_45_7	#7	400	#7	200	0.9	0.6	500000
Model 8	22	e2_8_1	8_12_14_12_7	#8	400	#8	200	0.9	0.6	500000
	23	e2_8_2	8_15_19_15_7	#8	400	#8	200	0.9	0.6	500000
	24	e2_8_3	8_45_7	#8	400	#8	200	0.9	0.6	500000

Table 3.2.3: Mean Utilization: Complex System: List of neural metamodels.

#	NNet	Data set	size	METHOD 1:			TOLERANCE approach		METHOD 2:		METHOD 3:	
				Alpha = 5%	Alpha = 6.5%	Alpha = 8%	MAD	(M_Std)	MMAD	(MM_Std)		
1	e2_1_1	TRAINING SET	600	0.67%	0.00%	0.00%	0.0082	0.0074	0.0185	0.0088		
		TEST SET	400	4.75%	1.25%	0.25%	0.0097	0.0094	0.0215	0.0123		
2	e2_1_2	TRAINING SET	600	0.33%	0.00%	0.00%	0.0074	0.0065	0.0164	0.0079		
		TEST SET	400	3.75%	1.00%	0.50%	0.0089	0.0091	0.0198	0.0127		
3	e2_1_3	TRAINING SET	600	0.67%	0.00%	0.00%	0.0094	0.0078	0.0204	0.0087		
		TEST SET	400	2.25%	0.75%	0.00%	0.0104	0.0090	0.0225	0.0103		
4	e2_2_1	TRAINING SET	600	0.17%	0.00%	0.00%	0.0053	0.0065	0.0162	0.0069		
		TEST SET	400	1.50%	0.50%	0.25%	0.0060	0.0079	0.0186	0.0096		
5	e2_2_2	TRAINING SET	600	0.17%	0.00%	0.00%	0.0051	0.0066	0.0158	0.0077		
		TEST SET	400	2.50%	0.50%	0.25%	0.0059	0.0084	0.0195	0.0113		
6	e2_2_3	TRAINING SET	600	0.17%	0.00%	0.00%	0.0036	0.0047	0.0111	0.0058		
		TEST SET	400	0.75%	0.50%	0.00%	0.0044	0.0063	0.0142	0.0090		
7	e2_3_1	TRAINING SET	600	0.00%	0.00%	0.00%	0.0033	0.0031	0.0080	0.0040		
		TEST SET	400	0.00%	0.00%	0.00%	0.0034	0.0032	0.0080	0.0041		
8	e2_3_2	TRAINING SET	600	0.00%	0.00%	0.00%	0.0027	0.0025	0.0062	0.0032		
		TEST SET	400	0.00%	0.00%	0.00%	0.0029	0.0028	0.0069	0.0038		
9	e2_3_3	TRAINING SET	600	0.00%	0.00%	0.00%	0.0051	0.0042	0.0110	0.0042		
		TEST SET	400	0.00%	0.00%	0.00%	0.0053	0.0043	0.0114	0.0046		
10	e2_4_1	TRAINING SET	600	0.00%	0.00%	0.00%	0.0022	0.0017	0.0047	0.0017		
		TEST SET	400	0.00%	0.00%	0.00%	0.0023	0.0019	0.0050	0.0019		
11	e2_4_2	TRAINING SET	600	0.00%	0.00%	0.00%	0.0018	0.0015	0.0039	0.0015		
		TEST SET	400	0.00%	0.00%	0.00%	0.0019	0.0016	0.0042	0.0019		
12	e2_4_3	TRAINING SET	600	0.00%	0.00%	0.00%	0.0021	0.0018	0.0045	0.0021		
		TEST SET	400	0.00%	0.00%	0.00%	0.0021	0.0018	0.0047	0.0021		
13	e2_5_1	TRAINING SET	400	0.25%	0.00%	0.00%	0.0077	0.0068	0.0168	0.0083		
		TEST SET	240	1.25%	1.25%	0.42%	0.0081	0.0077	0.0184	0.0104		
14	e2_5_2	TRAINING SET	400	0.00%	0.00%	0.00%	0.0051	0.0045	0.0113	0.0058		
		TEST SET	240	1.67%	1.25%	0.00%	0.0062	0.0066	0.0139	0.0107		
15	e2_5_3	TRAINING SET	400	0.25%	0.00%	0.00%	0.0076	0.0064	0.0161	0.0077		
		TEST SET	240	0.83%	0.83%	0.83%	0.0086	0.0073	0.0181	0.0098		
16	e2_6_1	TRAINING SET	400	11.75%	2.25%	0.50%	0.0167	0.0131	0.0350	0.0131		
		TEST SET	200	38.00%	20.50%	9.50%	0.0232	0.0192	0.0480	0.0215		
17	e2_6_2	TRAINING SET	400	9.25%	2.50%	0.75%	0.0155	0.0126	0.0333	0.0131		
		TEST SET	200	42.00%	18.00%	7.50%	0.0225	0.0196	0.0487	0.0232		
18	e2_6_3	TRAINING SET	400	12.00%	2.00%	0.75%	0.0169	0.0135	0.0345	0.0133		
		TEST SET	200	42.50%	22.50%	11.00%	0.0255	0.0208	0.0510	0.0233		

Table 3.2.4: Mean Utilization: Complex System: Results of metamodels

#	NNet	Data set	METHOD 4:		utilization				maximum utilization	
			%error	(%_Std)	min	mean	(U_Std)	max	av max util	(MU_std)
1	e2_1_1	TRAINING SET	1.266%	1.174%	29.600%	68.113%	17.120%	100.000%	86.063%	13.894%
		TEST SET	1.432%	1.356%	32.200%	68.963%	17.258%	100.000%	86.988%	13.694%
2	e2_1_2	TRAINING SET	1.127%	1.008%	29.600%	68.113%	17.120%	100.000%	86.063%	13.894%
		TEST SET	1.304%	1.253%	32.200%	68.963%	17.258%	100.000%	86.988%	13.694%
3	e2_1_3	TRAINING SET	1.467%	1.340%	29.600%	68.113%	17.120%	100.000%	86.063%	13.894%
		TEST SET	1.550%	1.328%	32.200%	68.963%	17.258%	100.000%	86.988%	13.694%
4	e2_2_1	TRAINING SET	0.584%	0.720%	69.500%	94.588%	6.874%	100.000%	100.000%	0.000%
		TEST SET	0.660%	0.880%	70.300%	94.872%	6.669%	100.000%	100.000%	0.000%
5	e2_2_2	TRAINING SET	0.561%	0.731%	69.500%	94.588%	6.874%	100.000%	100.000%	0.000%
		TEST SET	0.642%	0.925%	70.300%	94.872%	6.669%	100.000%	100.000%	0.000%
6	e2_2_3	TRAINING SET	0.393%	0.526%	69.500%	94.588%	6.874%	100.000%	100.000%	0.000%
		TEST SET	0.479%	0.700%	70.300%	94.872%	6.669%	100.000%	100.000%	0.000%
7	e2_3_1	TRAINING SET	0.511%	0.475%	41.500%	65.611%	10.994%	100.000%	83.334%	7.455%
		TEST SET	0.517%	0.472%	42.800%	65.962%	11.047%	99.700%	83.956%	7.556%
8	e2_3_2	TRAINING SET	0.406%	0.360%	41.500%	65.611%	10.994%	100.000%	83.334%	7.455%
		TEST SET	0.440%	0.395%	42.800%	65.962%	11.047%	99.700%	83.956%	7.556%
9	e2_3_3	TRAINING SET	0.768%	0.613%	41.500%	65.611%	10.994%	100.000%	83.334%	7.455%
		TEST SET	0.793%	0.608%	42.800%	65.962%	11.047%	99.700%	83.956%	7.556%
10	e2_4_1	TRAINING SET	0.516%	0.404%	28.500%	41.891%	6.467%	63.300%	53.325%	3.188%
		TEST SET	0.549%	0.435%	29.300%	42.030%	6.499%	61.800%	53.565%	3.251%
11	e2_4_2	TRAINING SET	0.414%	0.326%	28.500%	41.891%	6.467%	63.300%	53.325%	3.188%
		TEST SET	0.445%	0.361%	29.300%	42.030%	6.499%	61.800%	53.565%	3.251%
12	e2_4_3	TRAINING SET	0.491%	0.430%	28.500%	41.891%	6.467%	63.300%	53.325%	3.188%
		TEST SET	0.507%	0.422%	29.300%	42.030%	6.499%	61.800%	53.565%	3.251%
13	e2_5_1	TRAINING SET	1.277%	1.214%	29.600%	62.330%	14.526%	97.900%	79.159%	12.125%
		TEST SET	1.321%	1.255%	32.200%	62.714%	14.593%	97.800%	79.295%	12.089%
14	e2_5_2	TRAINING SET	0.850%	0.808%	29.600%	62.330%	14.526%	97.900%	79.159%	12.125%
		TEST SET	0.977%	0.963%	32.200%	62.714%	14.593%	97.800%	79.295%	12.089%
15	e2_5_3	TRAINING SET	1.280%	1.162%	29.600%	62.330%	14.526%	97.900%	79.159%	12.125%
		TEST SET	1.400%	1.134%	32.200%	62.714%	14.593%	97.800%	79.295%	12.089%
16	e2_6_1	TRAINING SET	2.641%	2.187%	30.160%	67.605%	17.254%	100.000%	85.693%	14.358%
		TEST SET	3.576%	3.109%	32.030%	69.157%	16.986%	100.000%	87.315%	13.145%
17	e2_6_2	TRAINING SET	2.472%	2.133%	30.160%	67.605%	17.254%	100.000%	85.693%	14.358%
		TEST SET	3.433%	3.017%	32.030%	69.157%	16.986%	100.000%	87.315%	13.145%
18	e2_6_3	TRAINING SET	2.705%	2.388%	30.160%	67.605%	17.254%	100.000%	85.693%	14.358%
		TEST SET	3.925%	3.322%	32.030%	69.157%	16.986%	100.000%	87.315%	13.145%

Table 3.2.4 (cont'd): Mean Utilization: Complex System: Results of metamodels

#	NNet	Data set	size	METHOD 1:			TOLERANCE approach		METHOD 2:		METHOD 3:	
				Alpha = 5%	Alpha = 6.5%	Alpha = 8%	MAD	(M_Std)	MMAD	(MM_Std)		
19	e2_7_1	TRAINING SET	400	25.25%	9.00%	3.00%	0.0186	0.0155	0.0422	0.0165		
		TEST SET	200	47.50%	18.50%	10.00%	0.0228	0.0189	0.0510	0.0195		
20	e2_7_2	TRAINING SET	400	21.75%	6.75%	2.00%	0.0178	0.0149	0.0410	0.0151		
		TEST SET	200	50.00%	27.00%	10.50%	0.0237	0.0199	0.0536	0.0210		
21	e2_7_3	TRAINING SET	400	24.00%	6.50%	2.25%	0.0183	0.0148	0.0411	0.0149		
		TEST SET	200	50.50%	26.50%	13.00%	0.0242	0.0208	0.0551	0.0226		
22	e2_8_1	TRAINING SET	400	41.50%	18.50%	6.25%	0.0220	0.0182	0.0492	0.0180		
		TEST SET	200	60.00%	38.50%	21.00%	0.0286	0.0245	0.0621	0.0261		
23	e2_8_2	TRAINING SET	400	35.75%	10.75%	2.00%	0.0202	0.0168	0.0449	0.0161		
		TEST SET	200	65.50%	43.00%	24.00%	0.0303	0.0281	0.0674	0.0332		
24	e2_8_3	TRAINING SET	400	33.75%	12.00%	2.00%	0.0206	0.0166	0.0447	0.0156		
		TEST SET	200	66.50%	44.50%	29.50%	0.0305	0.0259	0.0669	0.0296		

Table 3.2.4 (cont'd): Mean Utilization: Complex System: Results of metamodels

#	NNet	Data set	METHOD 4:		utilization				maximum utilization	
			%error	(%_Std)	min	mean	(U_Std)	max	av max util	(MU_std)
19	e2_7_1	TRAINING SET	2.934%	2.568%	30.280%	67.581%	17.208%	100.000%	85.518%	14.118%
		TEST SET	3.499%	2.984%	32.670%	69.236%	17.064%	100.000%	87.174%	13.315%
20	e2_7_2	TRAINING SET	2.801%	2.397%	30.280%	67.581%	17.208%	100.000%	85.518%	14.118%
		TEST SET	3.601%	3.042%	32.670%	69.236%	17.064%	100.000%	87.174%	13.315%
21	e2_7_3	TRAINING SET	2.927%	2.555%	30.280%	67.581%	17.208%	100.000%	85.518%	14.118%
		TEST SET	3.719%	3.288%	32.670%	69.236%	17.064%	100.000%	87.174%	13.315%
22	e2_8_1	TRAINING SET	3.459%	3.050%	29.940%	67.535%	17.354%	100.000%	85.441%	14.285%
		TEST SET	4.378%	3.882%	32.820%	69.042%	17.053%	100.000%	87.184%	13.228%
23	e2_8_2	TRAINING SET	3.269%	3.108%	29.940%	67.535%	17.354%	100.000%	85.441%	14.285%
		TEST SET	4.666%	4.467%	32.820%	69.042%	17.053%	100.000%	87.184%	13.228%
24	e2_8_3	TRAINING SET	3.323%	3.032%	29.940%	67.535%	17.354%	100.000%	85.441%	14.285%
		TEST SET	4.665%	4.076%	32.820%	69.042%	17.053%	100.000%	87.184%	13.228%

Table 3.2.4 (cont'd): Mean Utilization: Complex System: Results of metamodels

Mod #	Training set	Test set	Interarrival time		Processing time range	Due date tightness factor	Scheduling rule
			Nature	Range			
1	#1	#1	deterministic	[10..85]	deterministic	[2..9]	SPT, MOD or EDD
2	#2	#2	deterministic	[15..30]	deterministic	[6..9]	SPT, MOD or EDD
3	#3	#3	deterministic	[60..85]	deterministic	[2..5]	SPT, MOD or EDD
4	#4	#4	deterministic	[15..30]	deterministic	9	SPT, MOD or EDD
5	#5	#5	deterministic	[15..30]	deterministic	6	SPT, MOD or EDD
6	#6	#6	deterministic	[15..30]	deterministic	[6..9]	SPT
7	#7	#7	deterministic	[15..30]	deterministic	[6..9]	EDD
8	#8	#8	deterministic	[15..30]	deterministic	[6..9]	MOD
9	#9	#9	deterministic	[15..30]	deterministic	9	MOD
10	#10	#10	deterministic	[15..30]	deterministic	6	MOD
11	#11	#11	stochastic (exponential)	mean in [60..85]	deterministic	[2..5]	SPT, MOD or EDD
12	#12	#12	deterministic	[60..85]	stochastic (exponential)	[2..5]	SPT, MOD or EDD
13	#13	#13	stochastic (exponential)	mean in [60..85]	stochastic (exponential)	[2..5]	SPT, MOD or EDD
14	#14	#14	deterministic *machine utilization < 99%	[15..30]	deterministic	[6..9]	SPT, MOD or EDD
15	#15	#15	deterministic	[30..60]	deterministic	[2..5]	SPT, MOD or EDD
16	#16	#16	stochastic (exponential)	mean in [30..60]	deterministic	[2..5]	SPT, MOD or EDD
17	#17	#17	deterministic	[30..60]	stochastic (exponential)	[2..5]	SPT, MOD or EDD
18	#18	#18	stochastic (exponential)	mean in [30..60]	stochastic (exponential)	[2..5]	SPT, MOD or EDD

Table 4.1.1: Mean Tardiness: Simple System: List of models.

MOD #	Percentage of Tardy jobs				Machine utilization				Tardiness			
	min	mean	(std)	max.	min	mean	(std)	max.	min	mean	(std)	max.
1	0.0%	24.6%	31.6%	100.0%	16.0%	50.9%	21.6%	100.0%	0.00	37.76	130.09	1174.41
2	0.0%	27.7%	34.3%	100.0%	46.0%	81.3%	14.8%	100.0%	0.00	65.43	125.94	767.57
3	0.0%	23.0%	13.2%	81.7%	16.0%	24.9%	4.5%	34.0%	0.00	9.83	7.88	64.87
4	0.0%	22.9%	31.8%	100.0%	45.0%	82.4%	14.4%	100.0%	0.00	53.59	111.30	680.33
5	0.0%	37.1%	35.7%	100.0%	45.0%	82.3%	14.3%	100.0%	0.00	77.38	140.58	803.37
6	0.0%	12.6%	18.8%	74.4%	45.0%	82.0%	15.0%	100.0%	0.00	39.33	80.71	562.19
7	0.0%	36.6%	38.5%	100.0%	46.0%	80.9%	14.7%	100.0%	0.00	80.57	146.84	805.20
8	0.0%	35.3%	37.4%	100.0%	44.0%	80.4%	14.8%	100.0%	0.00	81.39	148.63	855.86
9	0.0%	35.5%	39.0%	100.0%	44.0%	81.9%	15.0%	100.0%	0.00	67.51	140.04	801.91
10	0.0%	44.4%	39.2%	100.0%	44.0%	80.8%	14.9%	100.0%	0.00	105.88	171.73	925.95
11	0.0%	22.5%	12.2%	63.8%	12.0%	24.8%	4.9%	39.0%	0.60	11.62	7.06	35.00
12	1.6%	29.0%	13.2%	89.9%	13.0%	25.1%	5.0%	39.0%	1.70	14.34	7.76	48.80
13	11.4%	34.6%	11.6%	65.8%	10.0%	26.0%	5.3%	40.0%	5.20	21.33	9.69	61.10
14	0.0%	12.8%	20.4%	92.5%	44.0%	77.4%	13.5%	98.0%	0.00	14.67	35.60	296.84
15	0.0%	27.8%	21.1%	84.8%	22.0%	41.8%	8.8%	69.0%	0.00	8.25	7.77	33.41
16	2.6%	33.1%	20.7%	82.9%	20.0%	41.7%	9.4%	70.0%	0.60	13.08	9.53	51.20
17	9.6%	41.7%	17.6%	87.8%	20.0%	41.8%	9.5%	70.0%	3.05	18.56	10.96	61.10
18	17.3%	49.9%	16.4%	89.6%	20.0%	41.8%	9.4%	70.0%	5.75	29.26	14.78	88.10

Table 4.1.2: Mean Tardiness: Simple System: Generated set characteristics.

NNet	Size	Training set #	(size)	Test set #	(size)	learning coef	momentum coef	# examples learned
------	------	----------------	--------	------------	--------	---------------	---------------	--------------------

Model 1	1	e3_1_1	5_9_10_8_3	#1	700	#1	400	0.9	0.6	500000
	2	e3_1_2	5_9_10_8_3	#1	700	#1	400	1.2	0.6	393704
	3	e3_1_3	5_10_11_9_3	#1	700	#1	400	0.9	0.6	300909
	4	e3_1_4	5_10_11_9_3	#1	700	#1	400	0.9	0.2	525913
	5	e3_1_5	5_10_11_9_3	#1	700	#1	400	0.9	0.2	481282
	6	e3_1_6	5_10_15_10_3*	#1	700	#1	400	0.9	0.6	137732
	7	e3_1_7	5_25_3*	#1	700	#1	400	0.9	0.6	366187
	8	e3_1_8	5_25_3	#1	700	#1	400	0.9	0.6	403657
	9	e3_1_9	5_40_3	#1	700	#1	400	0.9	0.6	263456
	10	e3_1_10	5_9_10_8_3	#1	700	#1	400	1.2	0.6	500000
	11	e3_1_11	5_9_10_8_3	#1	700	#1	400	1.2	0.0	500000
	12	e3_1_12	5_6_7_6_3	#1	700	#1	400	1.2	0.6	500000
	13	e3_1_13	5_40_3	#1	700	#1	400	0.9	0.6	500000
Model 2	14	e3_2_1	5_40_3	#2	500	#2	300	0.9	0.6	500000
	15	e3_2_2	5_40_3	#2	500	#2	300	1.2	0.6	500000
	16	e3_2_3	5_6_7_6_3	#2	500	#2	300	1.2	0.6	500000
	17	e3_2_4	5_55_3	#2	500	#2	300	0.9	0.5	530627
Model 3	18	e3_3_1	5_6_7_6_3	#3	500	#3	300	0.9	0.6	542911
	19	e3_3_2	5_6_7_6_3	#3	500	#3	300	1.2	0.2	500000
	20	e3_3_3	5_8_9_8_3	#3	500	#3	300	0.9	0.6	500000
	21	e3_3_4	5_42_3	#3	500	#3	300	0.9	0.6	574158
	22	e3_3_5	5_55_3	#3	500	#3	300	0.9	0.6	500000
	23	e3_3_6	5_11_14_11_3	#3	500	#3	300	0.9	0.6	524199
Model 4	24	e3_4_1	5_55_3	#4	500	#4	300	0.9	0.6	500000
	25	e3_4_2	5_6_7_6_3	#4	500	#4	300	0.9	0.6	500000
	26	e3_4_3	5_8_9_8_3	#4	500	#4	300	0.9	0.6	500000
Model 5	27	e3_5_1	5_55_3	#5	500	#5	300	0.9	0.6	500000
	28	e3_5_2	5_6_7_6_3	#5	500	#5	300	0.9	0.6	500000
	29	e3_5_3	5_8_9_8_3	#5	500	#5	300	0.9	0.6	500000
	30	e3_5_4	5_55_3	#5	500	#5	300	0.2	0.1	620094
	31	e3_5_5	5_55_3	#5	500	#5	300	1.0	0.0	563251
	32	e3_5_6	4_8_9_8_3	#5	500	#5	300	0.9	0.6	710110
Model 6	33	e3_6_1	5_6_7_6_3	#6	500	#6	300	0.9	0.6	500000
	34	e3_6_2	5_55_3	#6	500	#6	300	2	1	500000
	35	e3_6_3	5_8_9_8_3	#6	500	#6	300	0.9	0.6	500000
	36	e3_6_4	5_8_9_8_3	#6	500	#6	300	0.6	0.2	500000
Model 7	37	e3_7_1	5_6_7_6_3	#7	500	#7	300	0.9	0.6	500000
	38	e3_7_2	5_55_3	#7	500	#7	300	0.2	0.1	500000
	39	e3_7_3	5_8_9_8_3	#7	500	#7	300	0.9	0.6	550001
	40	e3_7_4	5_55_3	#7	500	#7	300	0.9	0.6	500000

Table 4.1.3: Mean Tardiness: Simple System: List of neural metamodels.

	NNet	Size	Training set #	(size)	Test set #	(size)	learning coef	momentum coef	examples learned	
Model 8	41	e3_8_1	5_6_7_6_3	#8	500	#8	300	0.9	0.6	500000
	42	e3_8_2	5_55_3	#8	500	#8	300	0.9	0.2	500000
	43	e3_8_3	5_8_9_8_3	#8	500	#8	300	0.9	0.6	500000
	44	e3_8_4	5_55_3	#8	500	#8	300	0.9	0.6	500000
Model 9	45	e3_9_1	5_6_7_6_3	#9	500	#9	300	0.9	0.6	500000
	46	e3_9_2	5_55_3	#9	500	#9	300	0.9	0.6	500000
	47	e3_9_3	5_8_9_8_3	#9	500	#9	300	0.9	0.6	500000
Model 10	48	e3_10_1	5_6_7_6_3	#10	500	#10	300	0.9	0.6	500000
	49	e3_10_2	5_55_3	#10	500	#10	300	0.9	0.6	500000
	50	e3_10_3	5_8_9_8_3	#10	500	#10	300	0.9	0.6	500000
Model 11	51	e3_11_1	5_6_7_6_3	#11	400	#11	200	0.9	0.6	500000
	52	e3_11_2	5_55_3	#11	400	#11	200	0.9	0.6	500000
	53	e3_11_3	5_8_9_8_3	#11	400	#11	200	0.9	0.6	500000
Model 12	54	e3_12_1	5_6_7_6_3	#12	400	#12	200	0.9	0.6	500000
	55	e3_12_2	5_55_3	#12	400	#12	200	0.9	0.6	500000
	56	e3_12_3	5_8_9_8_3	#12	400	#12	200	0.9	0.6	500000
Model 13	57	e3_13_1	5_6_7_6_3	#13	400	#13	200	0.9	0.6	500000
	58	e3_13_2	5_55_3	#13	400	#13	200	0.9	0.6	500000
	59	e3_13_3	5_8_9_8_3	#13	400	#13	200	0.9	0.6	500000
Model 14	60	e3_13_1	5_6_7_6_3	#14	332	#14	300	0.9	0.6	500000
	61	e3_13_2	5_55_3	#14	332	#14	300	0.9	0.6	500000
	62	e3_13_3	5_8_9_8_3	#14	332	#14	300	0.9	0.6	500000
Model 15	63	e3_13_1	5_6_7_6_3	#15	400	#15	300	0.9	0.6	500000
	64	e3_13_2	5_55_3	#15	400	#15	300	0.9	0.6	500000
	65	e3_13_3	5_8_9_8_3	#15	400	#15	300	0.9	0.6	500000
Model 16	66	e3_13_1	5_6_7_6_3	#16	400	#16	300	0.9	0.6	500000
	67	e3_13_2	5_55_3	#16	400	#16	300	0.9	0.6	500000
	68	e3_13_3	5_8_9_8_3	#16	400	#16	300	0.9	0.6	500000
Model 17	69	e3_13_1	5_6_7_6_3	#17	400	#17	300	0.9	0.6	500000
	70	e3_13_2	5_55_3	#17	400	#17	300	0.9	0.6	500000
	71	e3_13_3	5_8_9_8_3	#17	400	#17	300	0.9	0.6	500000
Model 18	72	e3_13_1	5_6_7_6_3	#18	400	#18	300	0.9	0.6	500000
	73	e3_13_2	5_55_3	#18	400	#18	300	0.9	0.6	500000
	74	e3_13_3	5_8_9_8_3	#18	400	#18	300	0.9	0.6	500000

* = Bias

Table 4.1.3 (Cont'd): Mean Tardiness: Simple System: List of neural metamodels.

#	NNet	Data set	size	METHOD 1:			Tolerance approach	METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	mean	(Std)	mean	(Std)	
1	e3_1_1	Training set	700	35.7%	18.9%	12.4%	10.3	19.5	15.9	25.7	
		Test set	400	38.3%	23.0%	17.3%	20.0	52.4	32.0	69.2	
2	e3_1_2	Training set	700	31.0%	19.0%	11.0%	8.6		17.7		
3	e3_1_3	Training set	700	39.0%	20.0%	14.0%	9.6		19.7		
4	e3_1_4	Training set	700	35.0%	21.0%	13.0%	8.8		18.6		
5	e3_1_5	Training set	700	35.0%	21.0%	11.0%	7.2		15.1		
6	e3_1_6	Training set	700	37.0%	25.0%	16.0%	14.4		33.0		
7	e3_1_7	Training set	700	40.0%	26.0%	16.0%	9.3		20.4		
8	e3_1_8	Training set	700	40.0%	24.0%	14.0%	9.3		20.3		
9	e3_1_9	Training set	700	45.0%	24.0%	14.0%	10.2		22.0		
10	e3_1_10	Training set	700	32.7%	16.0%	9.3%	9.4	18.0	15.4	24.8	
		Test set	400	34.0%	20.0%	16.5%	21.4	57.0	33.1	72.5	
11	e3_1_11	Training set	700	35.3%	19.6%	11.0%	9.3	17.9	14.8	24.6	
		Test set	400	36.0%	22.5%	17.3%	19.9	54.3	31.2	71.6	
12	e3_1_12	Training set	700	35.7%	22.1%	14.4%	12.6	29.9	20.2	41.9	
		Test set	400	36.0%	23.5%	17.3%	20.9	55.5	32.8	72.2	
13	e3_1_13	Training set	700	43.1%	27.1%	16.4%	12.8	26.9	20.7	35.8	
		Test set	400	41.8%	27.0%	19.8%	20.4	52.4	32.4	68.3	
14	e3_2_1	Training set	500	57.2%	37.2%	23.2%	11.2	17.9	21.8	24.2	
		Test set	300	60.7%	42.3%	31.0%	16.0	26.8	29.9	35.5	
15	e3_2_2	Training set	500	71.4%	50.4%	34.0%	15.5	21.7	28.7	27.4	
		Test set	300	75.7%	58.7%	40.3%	19.3	29.0	35.7	37.8	
16	e3_2_3	Training set	500	59.2%	39.6%	29.6%	13.6	20.7	24.3	26.4	
		Test set	300	65.0%	47.3%	34.3%	17.0	28.8	30.5	37.3	
17	e3_2_4	Training set	500	57.6%	33.6%	21.8%	10.4	15.3	19.2	19.3	
		Test set	300	63.7%	43.0%	31.3%	15.0	24.0	27.4	30.7	
18	e3_3_1	Training set	500	6.2%	2.0%	0.6%	2.5	3.1	4.5	4.4	
		Test set	300	8.7%	3.7%	1.3%	2.8	4.0	5.3	5.8	
19	e3_3_2	Training set	500	6.0%	2.0%	0.6%	2.5	3.1	4.5	4.4	
		Test set	300	9.7%	3.0%	1.3%	2.8	3.8	5.1	5.6	
20	e3_3_3	Training set	500	5.8%	1.8%	0.4%	2.4	3.0	4.3	4.3	
		Test set	300	10.0%	3.7%	1.3%	2.9	4.0	5.3	5.8	
21	e3_3_4	Training set	500	5.6%	1.2%	0.4%	2.4	2.9	4.3	3.9	
		Test set	300	10.0%	3.3%	1.0%	2.9	4.1	5.4	5.9	
22	e3_3_5	Training set	500	4.8%	1.4%	0.4%	2.4	2.9	4.4	4.0	
		Test set	300	11.0%	3.0%	1.0%	3.0	3.9	5.4	5.5	
23	e3_3_6	Training set	500	5.2%	1.2%	0.4%	2.4	2.8	4.4	3.8	
		Test set	300	11.3%	3.3%	1.3%	2.9	4.0	5.4	5.8	

Table 4.1.4: Mean Tardiness: Simple System: Results of metamodels

		METHOD 4: Deviation -to- flow time TARDINESS										
#	NNet	Data set	E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
1	e3_1_1	Training set	8%	9%	6%	8%	4%	6%	0.0	30.8	112.9	1073.7
		Test set	9%	13%	8%	13%	6%	11%	0.0	44.7	147.3	1275.1
10	e3_1_10	Training set	7%	8%	6%	8%	4%	6%	0.0	30.8	112.9	1073.7
		Test set	9%	13%	8%	13%	6%	10%	0.0	44.7	147.3	1275.1
11	e3_1_11	Training set	7%	9%	7%	9%	5%	6%	0.0	30.8	112.9	1073.7
		Test set	9%	17%	9%	20%	7%	14%	0.0	44.7	147.3	1275.1
12	e3_1_12	Training set	9%	17%	7%	9%	5%	7%	0.0	30.8	112.9	1073.7
		Test set	10%	18%	8%	12%	6%	10%	0.0	44.7	147.3	1275.1
13	e3_1_13	Training set	9%	13%	8%	11%	6%	8%	0.0	30.8	112.9	1073.7
		Test set	10%	15%	8%	13%	6%	10%	0.0	44.7	147.3	1275.1
14	e3_2_1	Training set	4%	4%	2%	3%	1%	2%	0.0	59.2	115.0	711.1
		Test set	4%	5%	2%	5%	2%	4%	0.0	71.7	136.9	824.1
15	e3_2_2	Training set	6%	6%	2%	3%	1%	2%	0.0	59.2	115.0	711.1
		Test set	6%	7%	2%	5%	2%	4%	0.0	71.7	136.9	824.1
16	e3_2_3	Training set	6%	5%	2%	4%	1%	3%	0.0	59.2	115.0	711.1
		Test set	6%	6%	2%	4%	2%	4%	0.0	71.7	136.9	824.1
17	e3_2_4	Training set	6%	8%	3%	8%	2%	5%	0.0	59.2	115.0	711.1
		Test set	4%	5%	2%	5%	2%	4%	0.0	71.7	136.9	824.1
18	e3_3_1	Training set	4%	4%	4%	5%	2%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	4%	7%	3%	5%	0.0	9.6	7.9	61.4
19	e3_3_2	Training set	4%	4%	4%	5%	3%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	4%	7%	3%	5%	0.0	9.6	7.9	61.4
20	e3_3_3	Training set	4%	4%	4%	4%	2%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	5%	7%	3%	5%	0.0	9.6	7.9	61.4
21	e3_3_4	Training set	3%	4%	4%	4%	3%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	5%	7%	3%	5%	0.0	9.6	7.9	61.4
22	e3_3_5	Training set	4%	4%	4%	4%	2%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	5%	7%	3%	5%	0.0	9.6	7.9	61.4
23	e3_3_6	Training set	4%	4%	4%	4%	2%	3%	0.0	10.0	7.9	68.3
		Test set	4%	5%	5%	8%	3%	5%	0.0	9.6	7.9	61.4

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

#	NNet	Data set	size	METHOD 1: Tolerance approach			METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	mean	(Std)	mean	(Std)
24	e3_4_1	Training set	500	50.4%	31.6%	16.6%	8.6	14.0	16.5	18.6
		Test set	300	50.7%	34.7%	22.0%	10.9	20.5	20.1	27.8
25	e3_4_2	Training set	500	47.4%	31.2%	19.2%	8.6	15.0	17.2	20.6
		Test set	300	49.3%	34.3%	27.3%	11.8	22.8	21.7	31.0
26	e3_4_3	Training set	500	46.6%	33.2%	21.8%	9.1	15.5	17.5	20.6
		Test set	300	50.7%	36.7%	25.3%	11.9	21.8	21.3	28.4
27	e3_5_1	Training set	500	64.4%	42.4%	30.0%	12.4	17.2	23.0	21.8
		Test set	300	60.0%	42.3%	29.7%	13.1	20.5	24.2	27.5
28	e3_5_2	Training set	500	72.2%	52.2%	38.4%	16.9	21.1	28.8	25.3
		Test set	300	66.3%	46.0%	37.3%	18.2	24.1	29.4	28.7
29	e3_5_3	Training set	500	64.4%	43.2%	28.2%	12.0	17.4	22.8	22.0
		Test set	300	59.3%	46.0%	32.7%	13.0	19.6	24.8	25.6
30	e3_5_4	Training set	500	67.2%	44.0%	30.2%	13.8	18.9	24.5	24.9
		Test set	300	62.3%	41.7%	28.3%	13.1	17.6	22.4	22.2
31	e3_5_5	Training set	500	62.8%	39.4%	23.2%	12.0	18.9	22.6	24.9
		Test set	300	54.0%	39.7%	28.0%	12.3	19.5	21.8	24.9
32	e3_5_6	Training set	500	62.2%	39.8%	26.2%	12.4	18.7	22.5	23.2
		Test set	300	57.0%	40.7%	29.7%	13.3	21.1	23.6	26.1
33	e3_6_1	Training set	500	67.0%	44.4%	33.4%	9.8	23.2	29.0	32.5
		Test set	500	65.2%	46.0%	34.8%	9.8	22.8	29.2	31.5
34	e3_6_2	Training set	500	72.4%	52.2%	35.8%	9.8	21.0	29.1	27.6
		Test set	300	77.7%	52.7%	35.0%	9.6	19.7	28.5	25.2
35	e3_6_3	Training set	500	60.6%	42.4%	28.8%	8.3	19.0	24.6	26.1
		Test set	300	63.3%	49.0%	35.3%	9.0	19.6	26.7	26.1
36	e3_6_4	Training set	500	61.2%	43.0%	31.0%	8.8	20.9	26.1	29.3
		Test set	300	62.7%	48.3%	34.3%	9.2	20.8	27.4	28.4
37	e3_7_1	Training set	500	49.0%	29.8%	15.4%	10.3	13.0	15.2	16.1
		Test set	300	45.0%	31.0%	18.7%	11.3	15.6	16.2	18.8
38	e3_7_2	Training set	500	59.8%	35.2%	22.2%	12.7	15.1	18.8	17.8
		Test set	300	57.0%	36.3%	24.3%	13.3	16.8	19.3	19.5
39	e3_7_3	Training set	500	46.8%	30.2%	16.4%	10.2	13.2	15.6	16.2
		Test set	300	47.3%	33.7%	20.7%	12.6	18.4	18.5	22.0
40	e3_7_4	Training set	500	53.2%	31.4%	19.4%	11.1	13.9	16.4	16.5
		Test set	300	48.7%	32.7%	19.7%	11.4	16.0	16.9	19.2

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

#	NNet	Data set	METHOD 4: deviation -to- flow time							TARDINESS		
			E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
24	e3_4_1	Training set	3%	4%	1%	3%	1%	2%	0.0	49.5	103.3	662.4
		Test set	3%	4%	1%	3%	1%	3%	0.0	57.7	119.3	698.3
25	e3_4_2	Training set	2%	4%	1%	3%	1%	2%	0.0	49.5	103.3	662.4
		Test set	3%	5%	1%	3%	1%	3%	0.0	57.7	119.3	698.3
26	e3_4_3	Training set	3%	4%	1%	3%	1%	2%	0.0	49.5	103.3	662.4
		Test set	3%	5%	2%	4%	1%	3%	0.0	57.7	119.3	698.3
27	e3_5_1	Training set	5%	5%	2%	3%	2%	2%	0.0	76.5	136.5	785.9
		Test set	4%	5%	2%	3%	2%	3%	0.0	78.3	144.6	820.9
28	e3_5_2	Training set	7%	8%	4%	4%	3%	3%	0.0	76.5	136.5	785.9
		Test set	8%	9%	4%	4%	3%	3%	0.0	78.3	144.6	820.9
29	e3_5_3	Training set	4%	5%	2%	3%	2%	3%	0.0	76.5	136.5	785.9
		Test set	4%	5%	2%	3%	2%	2%	0.0	78.3	144.6	820.9
30	e3_5_4	Training set	6%	6%	3%	4%	2%	3%	0.0	76.5	136.5	785.9
		Test set	5%	5%	3%	4%	2%	3%	0.0	78.3	144.6	820.9
31	e3_5_5	Training set	4%	5%	2%	3%	2%	2%	0.0	76.5	136.5	785.9
		Test set	4%	5%	2%	3%	1%	2%	0.0	78.3	144.6	820.9
32	e3_5_6	Training set	5%	5%	2%	4%	2%	3%	0.0	76.5	136.5	785.9
		Test set	4%	5%	2%	3%	2%	3%	0.0	78.3	144.6	820.9
33	e3_6_1	Training set	0%	0%	0%	0%	0%	0%	0.0	38.9	80.7	562.2
		Test set	0%	0%	0%	0%	0%	0%	0.0	39.7	80.7	562.2
34	e3_6_2	Training set	0%	0%	0%	0%	0%	0%	0.0	38.9	80.7	562.2
		Test set	0%	0%	0%	0%	0%	0%	0.0	39.5	79.0	489.2
35	e3_6_3	Training set	0%	0%	0%	0%	0%	0%	0.0	38.9	80.7	562.2
		Test set	0%	0%	0%	0%	0%	0%	0.0	39.5	79.0	489.2
36	e3_6_4	Training set	0%	0%	0%	0%	0%	0%	0.0	38.9	80.7	562.2
		Test set	0%	0%	0%	0%	0%	0%	0.0	39.5	79.0	489.2
37	e3_7_1	Training set	4%	4%	3%	4%	2%	3%	0.0	78.9	139.9	783.0
		Test set	4%	4%	3%	4%	2%	3%	0.0	82.2	153.8	827.4
38	e3_7_2	Training set	5%	6%	3%	4%	2%	3%	0.0	78.9	139.9	783.0
		Test set	6%	6%	3%	4%	2%	3%	0.0	82.2	153.8	827.4
39	e3_7_3	Training set	4%	4%	2%	4%	2%	3%	0.0	78.9	139.9	783.0
		Test set	4%	5%	3%	5%	2%	4%	0.0	82.2	153.8	827.4
40	e3_7_4	Training set	4%	5%	2%	4%	2%	3%	0.0	78.9	139.9	783.0
		Test set	4%	5%	3%	4%	2%	3%	0.0	82.2	153.8	827.4

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

#	NNet	Data set	size	METHOD 1:			Tolerance approach	METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	mean	(Std)	mean	(Std)	
41	e3_8_1	Training set	500	49.0%	32.2%	19.4%	10.2	12.9	15.7	15.6	
		Test set	300	46.7%	33.7%	20.0%	11.6	15.7	17.4	19.1	
42	e3_8_2	Training set	500	55.2%	34.2%	18.6%	11.2	13.8	16.7	16.5	
		Test set	300	55.3%	35.7%	22.0%	13.0	17.3	19.2	20.3	
43	e3_8_3	Training set	500	48.8%	30.2%	14.6%	9.4	12.7	15.0	15.8	
		Test set	300	47.7%	31.0%	22.0%	10.7	15.8	16.7	19.1	
44	e3_8_4	Training set	500	51.4%	30.8%	15.0%	9.7	12.7	14.8	15.0	
		Test set	300	52.0%	30.3%	18.0%	11.1	16.1	16.7	18.8	
45	e3_9_1	Training set	500	40.2%	25.0%	14.4%	8.2	12.3	12.9	15.5	
		Test set	300	39.0%	24.7%	14.3%	8.8	14.6	13.6	17.8	
46	e3_9_2	Training set	500	45.4%	25.2%	14.6%	8.8	13.7	13.9	17.0	
		Test set	300	45.7%	28.3%	15.0%	9.5	15.3	14.6	18.7	
47	e3_9_3	Training set	500	36.4%	23.2%	13.8%	8.0	13.3	12.6	16.7	
		Test set	300	36.3%	22.0%	12.7%	7.8	13.7	12.2	17.1	
48	e3_10_1	Training set	500	47.8%	30.6%	19.8%	11.5	14.3	16.6	17.0	
		Test set	300	52.5%	37.0%	25.5%	13.6	17.3	20.1	20.8	
49	e3_10_2	Training set	500	50.8%	32.0%	21.4%	11.2	15.7	16.6	18.6	
		Test set	300	55.0%	34.3%	25.3%	12.3	16.7	18.2	19.3	
50	e3_10_3	Training set	500	47.0%	33.2%	19.4%	11.6	16.8	17.3	20.2	
		Test set	300	51.8%	36.3%	24.5%	13.2	18.6	19.6	22.0	
51	e3_11_1	Training set	400	0.5%	0.0%	0.0%	1.8	1.5	3.0	1.7	
		Test set	200	0.0%	0.0%	0.0%	1.9	1.6	3.1	1.7	
52	e3_11_2	Training set	400	0.3%	0.0%	0.0%	1.7	1.5	2.9	1.6	
		Test set	200	0.0%	0.0%	0.0%	1.9	1.7	3.3	1.8	
53	e3_11_3	Training set	400	0.3%	0.0%	0.0%	1.8	1.5	3.0	1.7	
		Test set	200	0.0%	0.0%	0.0%	1.8	1.6	3.1	1.8	
51	e3_12_1	Training set	400	1.5%	0.5%	0.0%	1.9	1.9	3.2	2.4	
		Test set	200	5.0%	1.0%	0.0%	2.2	2.5	3.9	3.3	
52	e3_12_2	Training set	400	1.3%	0.5%	0.0%	1.9	1.8	3.1	2.3	
		Test set	200	5.0%	1.0%	0.0%	2.3	2.5	3.9	3.2	
53	e3_12_3	Training set	400	1.5%	0.5%	0.0%	1.8	1.8	3.1	2.3	
		Test set	200	5.5%	1.0%	0.0%	2.2	2.5	3.8	3.2	
57	e3_13_1	Training set	400	3.5%	0.5%	0.0%	2.9	2.5	4.9	2.8	
		Test set	200	5.0%	0.0%	0.0%	3.3	2.6	5.3	2.7	
58	e3_13_2	Training set	400	3.5%	0.0%	0.0%	2.8	2.4	4.7	2.5	
		Test set	200	6.0%	0.5%	0.0%	3.4	2.6	5.4	2.8	
58	e3_13_3	Training set	400	3.3%	0.0%	0.0%	2.8	2.3	4.7	2.5	
		Test set	200	5.0%	0.0%	0.0%	3.1	2.5	5.1	2.7	

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

#	NNet	Data set	METHOD 4: Deviation -to- flow time						TARDINESS			
			E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
41	e3_8_1	Training set	4%	4%	2%	3%	2%	2%	0.0	86.2	153.1	831.7
		Test set	4%	5%	3%	4%	2%	3%	0.0	76.6	144.2	880.0
42	e3_8_2	Training set	4%	5%	3%	3%	2%	3%	0.0	86.2	153.1	831.7
		Test set	5%	5%	3%	4%	2%	3%	0.0	76.6	144.2	880.0
43	e3_8_3	Training set	4%	4%	2%	3%	2%	3%	0.0	86.2	153.1	831.7
		Test set	4%	5%	3%	4%	2%	3%	0.0	76.6	144.2	880.0
44	e3_8_4	Training set	4%	4%	2%	3%	2%	3%	0.0	86.2	153.1	831.7
		Test set	4%	5%	2%	4%	2%	3%	0.0	76.6	144.2	880.0
45	e3_9_1	Training set	3%	3%	2%	3%	1%	2%	0.0	71.0	139.6	759.2
		Test set	3%	3%	2%	3%	1%	2%	0.0	64.0	140.5	844.7
46	e3_9_2	Training set	3%	4%	2%	3%	1%	2%	0.0	71.0	139.6	759.2
		Test set	3%	4%	2%	3%	1%	2%	0.0	64.0	140.5	844.7
47	e3_9_3	Training set	3%	3%	2%	3%	1%	3%	0.0	71.0	139.6	759.2
		Test set	2%	3%	2%	3%	1%	2%	0.0	64.0	140.5	844.7
48	e3_10_1	Training set	5%	5%	3%	3%	3%	3%	0.0	104.6	173.1	918.4
		Test set	5%	6%	4%	5%	3%	4%	0.0	107.2	170.4	933.5
49	e3_10_2	Training set	4%	5%	3%	4%	2%	3%	0.0	104.6	173.1	918.4
		Test set	5%	6%	3%	4%	2%	3%	0.0	107.2	170.4	933.5
50	e3_10_3	Training set	4%	5%	3%	4%	2%	3%	0.0	104.6	173.1	918.4
		Test set	5%	6%	3%	5%	2%	4%	0.0	107.2	170.4	933.5
51	e3_11_1	Training set	2%	2%	2%	2%	2%	1%	0.5	11.3	7.1	37.5
		Test set	2%	2%	2%	2%	2%	1%	0.7	11.9	7.0	32.5
52	e3_11_2	Training set	2%	2%	2%	2%	1%	1%	0.5	11.3	7.1	37.5
		Test set	2%	2%	3%	2%	2%	2%	0.7	11.9	7.0	32.5
53	e3_11_3	Training set	2%	2%	2%	2%	2%	1%	0.5	11.3	7.1	37.5
		Test set	2%	2%	2%	2%	2%	2%	0.7	11.9	7.0	32.5
54	e3_12_1	Training set	2%	2%	2%	3%	2%	2%	1.6	14.6	8.0	55.9
		Test set	3%	4%	3%	4%	2%	3%	1.8	14.1	7.5	41.7
55	e3_12_2	Training set	2%	2%	2%	3%	2%	2%	1.6	14.6	8.0	55.9
		Test set	3%	4%	3%	4%	2%	3%	1.8	14.1	7.5	41.7
56	e3_12_3	Training set	2%	2%	2%	3%	2%	2%	1.6	14.6	8.0	55.9
		Test set	3%	3%	3%	4%	2%	3%	1.8	14.1	7.5	41.7
57	e3_13_1	Training set	3%	3%	3%	2%	2%	2%	4.9	21.6	10.0	64.3
		Test set	4%	3%	3%	3%	2%	2%	5.5	21.1	9.4	57.9
58	e3_13_2	Training set	3%	3%	3%	2%	2%	2%	4.9	21.6	10.0	64.3
		Test set	4%	3%	3%	3%	3%	2%	5.5	21.1	9.4	57.9
59	e3_13_3	Training set	3%	3%	3%	2%	2%	2%	4.9	21.6	10.0	64.3
		Test set	3%	3%	3%	3%	2%	2%	5.5	21.1	9.4	57.9

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

#	NNet	Data set	size	METHOD 1:			Tolerance approach	METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	mean	(Std)	mean	(Std)	
60	e3_14_1	Training set	332	35.2%	21.4%	11.1%	5.9	11.0	11.4	14.9	
		Test set	300	39.3%	26.3%	19.0%	8.0	16.6	16.4	23.8	
61	e3_14_2	Training set	332	35.5%	17.2%	9.0%	4.7	8.4	10.3	11.3	
		Test set	300	43.0%	28.3%	20.3%	8.0	17.0	18.2	24.3	
62	e3_14_3	Training set	332	35.8%	22.6%	15.4%	7.6	13.7	13.4	17.8	
		Test set	300	39.7%	28.3%	18.7%	9.0	18.1	17.6	25.6	
63	e3_15_1	Training set	400	0.8%	0.5%	0.0%	1.4	1.6	2.3	2.2	
		Test set	300	1.3%	0.0%	0.0%	1.6	1.8	2.7	2.3	
64	e3_15_2	Training set	400	0.5%	0.3%	0.0%	1.4	1.6	2.4	2.1	
		Test set	300	2.0%	0.0%	0.0%	1.7	1.9	2.9	2.4	
65	e3_15_3	Training set	400	0.5%	0.5%	0.0%	1.4	1.6	2.4	2.1	
		Test set	300	1.3%	0.0%	0.0%	1.8	1.8	2.9	2.3	
66	e3_16_1	Training set	400	0.5%	0.0%	0.0%	1.4	1.3	2.4	1.6	
		Test set	300	0.3%	0.0%	0.0%	1.6	1.5	2.8	1.7	
67	e3_16_2	Training set	400	0.0%	0.0%	0.0%	1.4	1.2	2.3	1.4	
		Test set	300	0.3%	0.0%	0.0%	1.6	1.5	2.7	1.8	
68	e3_16_3	Training set	400	0.0%	0.0%	0.0%	1.4	1.3	2.4	1.5	
		Test set	300	0.7%	0.0%	0.0%	1.6	1.5	2.8	1.8	
69	e3_17_1	Training set	400	0.8%	0.0%	0.0%	1.9	1.8	3.1	2.0	
		Test set	300	1.7%	0.0%	0.0%	2.0	1.8	3.3	2.1	
70	e3_17_2	Training set	400	0.5%	0.0%	0.0%	1.9	1.7	3.1	1.9	
		Test set	300	0.3%	0.0%	0.0%	2.1	1.8	3.4	2.0	
71	e3_17_3	Training set	400	0.8%	0.0%	0.0%	1.8	1.7	3.0	1.9	
		Test set	300	1.0%	0.0%	0.0%	1.9	1.7	3.1	1.9	
72	e3_18_1	Training set	400	6.5%	0.0%	0.0%	3.0	2.6	5.0	2.9	
		Test set	300	8.7%	1.3%	0.0%	3.4	3.2	5.6	3.5	
73	e3_18_2	Training set	400	4.5%	0.0%	0.0%	2.8	2.4	4.7	2.6	
		Test set	300	9.7%	0.7%	0.0%	3.5	3.1	5.5	3.4	
74	e3_18_3	Training set	400	3.8%	0.0%	0.0%	2.9	2.5	4.8	2.7	
		Test set	300	9.7%	0.7%	0.0%	3.6	3.2	5.8	3.5	

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

		METHOD 4:							TARDINESS			
#	NNet	Data set	E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
60	e3_14_1	Training set	3%	4%	2%	4%	1%	3%	0.0	13.8	33.9	341.8
		Test set	3%	6%	2%	5%	1%	3%	0.0	15.6	37.3	251.9
61	e3_14_2	Training set	2%	3%	2%	3%	1%	2%	0.0	13.8	33.9	341.8
		Test set	3%	7%	2%	4%	1%	3%	0.0	15.6	37.3	251.9
62	e3_14_3	Training set	3%	5%	3%	5%	2%	4%	0.0	13.8	33.9	341.8
		Test set	3%	6%	2%	5%	2%	4%	0.0	15.6	37.3	251.9
63	e3_15_1	Training set	2%	2%	2%	3%	1%	2%	0.0	7.9	7.8	35.2
		Test set	2%	2%	3%	3%	2%	2%	0.0	8.6	7.7	31.6
64	e3_15_2	Training set	2%	2%	2%	3%	2%	2%	0.0	7.9	7.8	35.2
		Test set	2%	3%	3%	3%	2%	2%	0.0	8.6	7.7	31.6
65	e3_15_3	Training set	2%	2%	2%	3%	2%	2%	0.0	7.9	7.8	35.2
		Test set	2%	2%	3%	3%	2%	2%	0.0	8.6	7.7	31.6
66	e3_16_1	Training set	2%	1%	2%	1%	1%	1%	0.6	12.7	9.5	60.1
		Test set	2%	2%	2%	1%	1%	1%	0.6	13.5	9.6	42.3
67	e3_16_2	Training set	2%	1%	1%	1%	1%	1%	0.6	12.7	9.5	60.1
		Test set	2%	2%	2%	1%	1%	1%	0.6	13.5	9.6	42.3
68	e3_16_3	Training set	2%	1%	1%	1%	1%	1%	0.6	12.7	9.5	60.1
		Test set	2%	2%	2%	1%	1%	1%	0.6	13.5	9.6	42.3
69	e3_17_1	Training set	2%	2%	2%	2%	1%	1%	2.3	18.2	11.0	60.6
		Test set	3%	2%	2%	1%	1%	1%	3.8	18.9	10.9	61.6
70	e3_17_2	Training set	2%	2%	2%	2%	1%	1%	2.3	18.2	11.0	60.6
		Test set	3%	2%	2%	2%	1%	1%	3.8	18.9	10.9	61.6
71	e3_17_3	Training set	2%	2%	2%	2%	1%	1%	2.3	18.2	11.0	60.6
		Test set	2%	2%	2%	1%	1%	1%	3.8	18.9	10.9	61.6
72	e3_18_1	Training set	3%	3%	2%	2%	2%	1%	5.1	28.8	14.8	81.1
		Test set	3%	3%	3%	2%	2%	2%	6.4	29.7	14.7	95.1
73	e3_18_2	Training set	3%	2%	2%	2%	2%	1%	5.1	28.8	14.8	81.1
		Test set	3%	3%	3%	2%	2%	2%	6.4	29.7	14.7	95.1
74	e3_18_3	Training set	3%	2%	2%	2%	2%	1%	5.1	28.8	14.8	81.1
		Test set	3%	3%	3%	2%	2%	2%	6.4	29.7	14.7	95.1

Table 4.1.4 (cont'd): Mean Tardiness: Simple System: Results of metamodels

MOD #	Training set	Test set	# of job types	# of mach	Interarrival time		Processing time range	Due date tightness factor	Scheduling rule
					Nature	Range			
1	#1	#1	6	7	deterministic	[20..100]	deterministic	[2..9]	SPT, MODD or EDD
2	#2	#2	6	7	deterministic	[20..40]	deterministic	[2..9]	SPT, MODD or EDD
3	#3	#3	6	7	deterministic	[40..70]	deterministic	[2..9]	SPT, MODD or EDD
4	#4	#4	6	7	deterministic	[70..100]	deterministic	[2..9]	SPT, MODD or EDD
5	#5	#5	6	7	deterministic	[20..100] *machine utilization < 98%	deterministic	[2..9]	SPT, MODD or EDD
6	#6	#6	6	7	stochastic (exponential)	mean in [40..100]	deterministic	[2..9]	SPT, MODD or EDD
7	#7	#7	6	7	deterministic	[40..100]	stochastic (exponential)	[2..9]	SPT, MODD or EDD
8	#8	#8	6	7	stochastic (exponential)	mean in [40..100]	stochastic (exponential)	[2..9]	SPT, MODD or EDD

Table 4.2.1: Mean Tardiness: Complex System: List of models.

MOD #	Percentage of tardy jobs				Machine utilization				Tardiness		Max. Tardiness	
	min.	mean	(Std)	max.	min.	mean	(Std)	max.	mean	(Std)	mean	(Std)
1	0.0%	34.1%	37.1%	100.0%	30.0%	68.5%	17.5%	100.0%	116.2	295.7	246.9	409.5
2	0.0%	83.3%	29.7%	100.0%	69.5%	94.7%	6.8%	100.0%	1032.8	780.1	1458.7	785.0
3	0.0%	22.1%	30.0%	100.0%	41.5%	65.8%	11.0%	100.0%	16.6	37.1	42.2	63.5
4	0.0%	12.9%	19.9%	90.1%	30.0%	41.8%	7.3%	60.0%	5.5	10.5	9.2	15.1
5	0.0%	22.4%	29.9%	100.0%	29.6%	62.5%	14.6%	97.9%	19.3	45.7	44.4	75.6
6	0.0%	21.7%	27.5%	98.2%	29.2%	53.8%	11.2%	99.9%	17.9	42.5	41.4	76.5
7	0.0%	32.9%	27.7%	100.0%	28.7%	53.9%	11.3%	100.0%	28.0	47.6	49.1	67.5
8	0.0%	38.8%	28.3%	100.0%	29.8%	53.9%	11.4%	100.0%	42.3	66.6	75.4	94.0

Table 4.2.2: Mean Tardiness: Complex System: Generated set characteristics.

NNet		Size	Training set #	(size)	Test set #	(size)	learning coef	momentum coef	examples learned	
Model 1	1	a_1_1	8_12_14_12_6	#1	600	#1	400	0.9	0.6	500000
	2	a_1_2	8_15_19_15_6	#1	600	#1	400	0.9	0.6	500000
	3	a_1_3	8_45_6	#1	600	#1	400	0.9	0.6	500000
Model 2	4	a_2_1	8_12_14_12_6	#2	600	#2	400	0.9	0.6	500000
	5	a_2_2	8_15_19_15_6	#2	600	#2	400	0.9	0.6	500000
	6	a_2_3	8_45_6	#2	600	#2	400	0.9	0.6	500000
Model 3	7	a_3_1	8_12_14_12_6	#3	600	#3	400	0.9	0.6	500000
	8	a_3_2	8_15_19_15_6	#3	600	#3	400	0.9	0.6	500000
	9	a_3_3	8_45_6	#3	600	#3	400	0.9	0.6	500000
Model 4	10	a_4_1	8_12_14_12_6	#4	600	#4	400	0.9	0.6	500000
	11	a_4_2	8_15_19_15_6	#4	600	#4	400	0.9	0.6	500000
	12	a_4_3	8_45_6	#4	600	#4	400	0.9	0.6	500000
Model 5	13	a_5_1	8_12_14_12_6	#5	400	#5	240	0.9	0.6	500000
	14	a_5_2	8_15_19_15_6	#5	400	#5	240	0.9	0.6	500000
	15	a_5_3	8_45_6	#5	400	#5	240	0.9	0.6	500000
Model 6	16	a_6_1	8_12_14_12_6	#6	500	#6	300	0.9	0.6	500000
	17	a_6_2	8_15_19_15_6	#6	500	#6	300	0.9	0.6	500000
	18	a_6_3	8_45_6	#6	500	#6	300	0.9	0.6	500000
Model 7	19	a_7_1	8_12_14_12_6	#7	500	#7	300	0.9	0.6	500000
	20	a_7_2	8_15_19_15_6	#7	500	#7	300	0.9	0.6	500000
	21	a_7_3	8_45_6	#7	500	#7	300	0.9	0.6	500000
Model 8	22	a_8_1	8_12_14_12_6	#8	500	#8	300	0.9	0.6	500000
	23	a_8_2	8_15_19_15_6	#8	500	#8	300	0.9	0.6	500000
	24	a_8_3	8_45_6	#8	500	#8	300	0.9	0.6	500000

Table 4.2.3: Mean Tardiness: Complex System: List of neural metamodels.

#	NNet	Data set	# obs	METHOD 1:			TOLERANCE approach	METHOD 2:		METHOD 3:	
				Alpha = 5%	Alpha = 6.5%	Alpha = 8%	MAD	(M_Std)	MMAD	(MM_Std)	
1	e4_1_1	TRAINING SET	600	76.8%	63.3%	57.3%	31.2	57.0	73.7	92.9	
		TEST SET	400	80.5%	66.0%	60.0%	59.0	122.0	127.2	171.8	
2	e4_1_2	TRAINING SET	600	74.5%	60.2%	54.0%	22.2	35.8	54.0	57.3	
		TEST SET	400	75.0%	64.5%	57.0%	47.4	108.0	99.4	146.4	
3	e4_1_3	TRAINING SET	600	77.5%	66.2%	57.7%	22.8	36.7	57.6	57.5	
		TEST SET	400	81.5%	68.5%	62.8%	46.6	95.6	108.0	145.6	
4	e4_2_1	TRAINING SET	600	100.0%	99.7%	97.5%	48.8	51.6	112.7	75.4	
		TEST SET	400	100.0%	100.0%	99.0%	55.2	56.5	125.9	77.3	
5	e4_2_2	TRAINING SET	600	100.0%	99.8%	99.3%	46.1	50.8	110.9	75.2	
		TEST SET	400	100.0%	99.5%	97.8%	51.8	54.8	121.3	76.5	
6	e4_2_3	TRAINING SET	600	100.0%	100.0%	98.5%	48.0	49.5	109.9	70.5	
		TEST SET	400	100.0%	100.0%	99.0%	54.9	55.8	126.8	77.0	
7	e4_3_1	TRAINING SET	600	34.5%	15.5%	6.2%	3.4	6.9	10.0	12.4	
		TEST SET	400	41.3%	25.0%	18.0%	5.5	14.8	18.1	29.4	
8	e4_3_2	TRAINING SET	600	29.7%	11.0%	3.5%	2.9	5.7	8.4	10.0	
		TEST SET	400	41.3%	24.3%	17.3%	5.4	15.4	18.2	31.1	
9	e4_3_3	TRAINING SET	600	38.5%	14.0%	5.2%	3.3	6.1	10.0	10.4	
		TEST SET	400	45.8%	27.8%	17.8%	6.0	15.9	19.4	30.4	
10	e4_4_1	TRAINING SET	600	11.0%	3.7%	0.3%	1.6	3.4	3.5	5.8	
		TEST SET	400	12.3%	4.0%	0.5%	1.6	3.5	3.8	6.0	
11	e4_4_2	TRAINING SET	600	11.2%	3.7%	0.3%	1.6	3.4	3.5	5.8	
		TEST SET	400	11.5%	4.0%	0.5%	1.6	3.5	3.7	6.0	
12	e4_4_3	TRAINING SET	600	0.7%	0.0%	0.0%	1.0	1.5	2.2	2.4	
		TEST SET	400	8.0%	0.8%	0.0%	1.4	2.4	3.2	4.0	
13	e4_5_1	TRAINING SET	400	47.5%	28.3%	20.5%	6.5	13.0	18.3	22.9	
		TEST SET	240	52.1%	35.8%	30.0%	15.3	42.8	40.7	74.2	
14	e4_5_2	TRAINING SET	400	44.5%	24.8%	13.0%	5.3	9.9	14.2	16.6	
		TEST SET	240	49.2%	32.9%	26.7%	12.6	34.5	32.9	56.8	
15	e4_5_3	TRAINING SET	400	41.0%	15.8%	4.8%	3.9	6.4	10.4	10.0	
		TEST SET	240	52.9%	39.2%	29.2%	12.3	32.6	32.9	56.1	

Table 4.2.4: Mean Tardiness: Complex System: Results of metamodels

#	NNet	Data set	METHOD 4:										deviation-to-flow time approach	
			E4 - 1 (Std1)	E4 - 2 (Std2)	E4 - 3 (Std3)	E4 - 4 (Std4)	E4 - 5 (Std5)	E4 - 6 (Std6)						
1	e4_1_1	TRAINING SET	17%	39%	17%	48%	21%	48%	31%	48%	26%	42%	23%	47%
		TEST SET	33%	99%	34%	131%	46%	117%	55%	86%	46%	73%	40%	66%
2	e4_1_2	TRAINING SET	13%	27%	12%	24%	17%	33%	23%	33%	18%	25%	14%	20%
		TEST SET	32%	100%	33%	136%	39%	99%	43%	75%	37%	67%	32%	64%
3	e4_1_3	TRAINING SET	13%	19%	11%	19%	14%	25%	24%	33%	19%	26%	14%	20%
		TEST SET	30%	90%	30%	120%	36%	93%	44%	75%	42%	77%	31%	57%
4	e4_2_1	TRAINING SET	41%	44%	53%	62%	68%	81%	31%	31%	35%	46%	22%	22%
		TEST SET	53%	57%	60%	72%	80%	107%	36%	34%	40%	40%	25%	27%
5	e4_2_2	TRAINING SET	37%	45%	44%	51%	65%	83%	33%	34%	33%	42%	20%	21%
		TEST SET	48%	53%	55%	66%	79%	105%	36%	40%	36%	36%	23%	27%
6	e4_2_3	TRAINING SET	44%	45%	50%	58%	82%	84%	31%	31%	30%	40%	18%	18%
		TEST SET	54%	53%	58%	68%	95%	106%	36%	38%	35%	36%	21%	22%
7	e4_3_1	TRAINING SET	1%	2%	1%	2%	2%	3%	5%	8%	2%	4%	2%	4%
		TEST SET	2%	3%	2%	3%	3%	5%	9%	19%	4%	9%	3%	7%
8	e4_3_2	TRAINING SET	1%	2%	1%	2%	2%	3%	4%	7%	2%	4%	2%	4%
		TEST SET	1%	3%	2%	3%	3%	7%	10%	20%	4%	9%	3%	6%
9	e4_3_3	TRAINING SET	1%	1%	1%	2%	2%	3%	5%	7%	2%	3%	2%	3%
		TEST SET	2%	3%	2%	3%	3%	5%	10%	19%	4%	8%	3%	7%
10	e4_4_1	TRAINING SET	2%	3%	2%	4%	1%	2%	1%	3%	1%	2%	1%	3%
		TEST SET	2%	3%	2%	3%	1%	1%	1%	3%	1%	3%	1%	3%
11	e4_4_2	TRAINING SET	2%	3%	2%	4%	1%	2%	1%	3%	1%	2%	1%	3%
		TEST SET	2%	3%	2%	3%	1%	1%	1%	3%	1%	3%	1%	3%
12	e4_4_3	TRAINING SET	1%	2%	1%	2%	1%	1%	1%	1%	1%	1%	1%	1%
		TEST SET	2%	2%	2%	3%	1%	2%	1%	2%	1%	2%	1%	2%
13	e4_5_1	TRAINING SET	2%	2%	2%	2%	7%	16%	8%	14%	5%	7%	4%	7%
		TEST SET	6%	14%	5%	13%	10%	30%	19%	44%	13%	32%	11%	31%
14	e4_5_2	TRAINING SET	1%	2%	2%	2%	5%	9%	6%	10%	4%	7%	4%	8%
		TEST SET	6%	17%	5%	14%	11%	30%	15%	36%	10%	27%	9%	28%
15	e4_5_3	TRAINING SET	1%	2%	1%	2%	4%	7%	4%	6%	3%	4%	3%	4%
		TEST SET	5%	15%	4%	13%	11%	33%	15%	32%	10%	25%	9%	26%

Table 4.2.4 (cont'd): Mean Tardiness: Complex System: Results of metamodels

#	NNet	Data set	# obs	METHOD 1:			TOLERANCE approach	METHOD 2:		METHOD 3:	
				Alpha = 5%	Alpha = 6.5%	Alpha = 8%	MAD	(M_Std)	MMAD	(MM_Std)	
16	e4_6_1	TRAINING SET	500	42.4%	15.6%	6.8%	3.9	8.2	12.4	16.3	
		TEST SET	300	45.0%	25.3%	16.0%	7.1	26.0	22.5	54.5	
17	e4_6_2	TRAINING SET	500	38.0%	15.0%	6.4%	3.7	7.7	11.9	15.1	
		TEST SET	300	44.3%	25.3%	16.7%	7.7	30.4	25.7	64.9	
18	e4_6_3	TRAINING SET	500	33.4%	12.8%	5.4%	3.2	5.8	9.6	10.5	
		TEST SET	300	44.3%	26.3%	16.3%	6.7	22.7	20.1	38.9	
19	e4_7_1	TRAINING SET	500	32.4%	12.6%	5.2%	4.2	7.5	10.1	11.5	
		TEST SET	300	38.0%	19.7%	13.0%	7.7	20.9	17.2	31.0	
20	e4_7_2	TRAINING SET	500	30.8%	11.2%	5.0%	4.1	6.8	9.7	10.2	
		TEST SET	300	37.0%	18.7%	13.7%	7.8	20.6	17.4	30.5	
21	e4_7_3	TRAINING SET	500	32.6%	10.0%	4.0%	4.0	6.7	9.4	9.9	
		TEST SET	300	34.3%	20.0%	13.7%	7.5	20.2	16.6	29.8	
22	e4_8_1	TRAINING SET	500	54.6%	29.0%	15.0%	7.5	13.7	17.6	21.3	
		TEST SET	300	63.3%	37.7%	24.3%	14.4	39.2	32.1	58.7	
23	e4_8_2	TRAINING SET	500	52.0%	22.6%	8.8%	5.9	8.4	14.0	12.3	
		TEST SET	300	57.7%	32.7%	22.3%	13.2	37.3	30.0	57.2	
24	e4_8_3	TRAINING SET	500	73.6%	46.2%	35.4%	12.8	19.4	28.0	25.3	
		TEST SET	300	80.7%	51.0%	34.3%	16.8	31.1	36.6	42.8	

Table 4.2.4 (cont'd): Mean Tardiness: Complex System: Results of metamodels

#	NNet	Data set	METHOD 4:										deviation-to-flow time approach	
			E4 - 1	(Std1)	E4 - 2	(Std2)	E4 - 3	(Std3)	E4 - 4	(Std4)	E4 - 5	(Std5)	E4 - 6	(Std6)
16	e4_6_1	TRAINING SET	1%	2%	1%	2%	2%	4%	7%	10%	3%	4%	2%	3%
		TEST SET	2%	3%	2%	3%	3%	4%	11%	28%	5%	11%	4%	8%
17	e4_6_2	TRAINING SET	1%	2%	1%	1%	2%	4%	6%	9%	3%	4%	2%	3%
		TEST SET	2%	3%	2%	3%	3%	4%	12%	30%	5%	10%	4%	9%
18	e4_6_3	TRAINING SET	1%	1%	1%	1%	2%	3%	5%	7%	3%	3%	2%	2%
		TEST SET	2%	2%	2%	2%	3%	4%	9%	20%	5%	11%	4%	10%
19	e4_7_1	TRAINING SET	2%	3%	2%	3%	2%	2%	4%	6%	3%	5%	4%	6%
		TEST SET	3%	4%	3%	4%	2%	3%	8%	16%	6%	14%	7%	15%
20	e4_7_2	TRAINING SET	2%	3%	2%	3%	2%	2%	4%	6%	3%	5%	4%	5%
		TEST SET	3%	4%	3%	4%	2%	4%	8%	17%	7%	15%	7%	15%
21	e4_7_3	TRAINING SET	2%	3%	2%	3%	2%	2%	4%	5%	3%	5%	3%	5%
		TEST SET	3%	4%	3%	4%	2%	3%	7%	16%	6%	14%	7%	14%
22	e4_8_1	TRAINING SET	4%	5%	4%	4%	3%	4%	7%	11%	6%	10%	6%	10%
		TEST SET	5%	7%	5%	7%	4%	6%	14%	34%	13%	32%	13%	31%
23	e4_8_2	TRAINING SET	4%	5%	4%	4%	3%	3%	6%	7%	5%	6%	5%	6%
		TEST SET	5%	7%	5%	7%	4%	6%	12%	31%	11%	28%	12%	27%
24	e4_8_3	TRAINING SET	4%	4%	4%	4%	4%	4%	15%	16%	12%	14%	11%	12%
		TEST SET	5%	6%	5%	6%	5%	6%	18%	26%	15%	22%	14%	21%

Table 4.2.4 (cont'd): Mean Tardiness: Complex System: Results of metamodels

Mod #	Training set	Test set	Interarrival time Range	Initial # of parts per type	Due date tightness factor	Scheduling rule
1	#1	#1	[15..60] deterministic	[1..15]	[2..5]	SPT, MODD or EDD
2	#2	#2	[15..60]	[1..15]	[2..5]	SPT, MODD or EDD
3	#3	#3	[15..35]	[1..15]	[2..5]	SPT, MODD or EDD
4	#4	#4	[35..60]	[1..15]	[2..5]	SPT, MODD or EDD
5	#5	#5	[15..60]	[1..5]	[2..5]	SPT, MODD or EDD
6	#6	#6	[15..60]	[6..10]	[2..5]	SPT, MODD or EDD
7	#7	#7	[15..60]	[11..15]	[2..5]	SPT, MODD or EDD
8	#8	#8	[15..60]	[1..15]	[6..9]	SPT, MODD or EDD
9	#9	#9	[15..60]	5	[2..5]	SPT, MODD or EDD
10	#10	#10	[15..60]	10	[2..5]	SPT, MODD or EDD
11	#11	#11	[15..60]	15	[2..5]	SPT, MODD or EDD
12	#12	#12	[15..60]	20	[2..5]	SPT, MODD or EDD
13	#13	#13	[15..60]	25	[2..5]	SPT, MODD or EDD
14	#14	#14	[15..60]	30	[2..5]	SPT, MODD or EDD
15	#15	#15	[15..60]	[3..7]	[2..5]	SPT, MODD or EDD
16	#16	#16	[15..60]	[8..12]	[2..5]	SPT, MODD or EDD
17	#17	#17	[15..60]	[13..17]	[2..5]	SPT, MODD or EDD
18	#18	#18	[35..60]	5	[2..5]	SPT, MODD or EDD
19	#19	#19	[35..60]	10	[2..5]	SPT, MODD or EDD
20	#20	#20	[35..60]	15	[2..5]	SPT, MODD or EDD
21	#21	#21	[35..60]	20	[2..5]	SPT, MODD or EDD
22	#22	#22	[15..35]	10	[2..5]	SPT, MODD or EDD
23	#23	#23	[15..35]	15	[2..5]	SPT, MODD or EDD
24	#24	#24	[15..35]	20	[2..5]	SPT, MODD or EDD
25	#25	#25	[15..35]	25	[2..5]	SPT, MODD or EDD

Table 5.1: Mean Tardiness: short term estimation: List of models.

MOD #	Percentage of Tardy jobs				Machine utilization				Tardiness			
	min	mean	(std)	max.	min	mean	(std)	max.	min	mean	(std)	max.
1	10.1%	78.9%	18.7%	100.0%	33.0%	74.7%	15.4%	100.0%	3.90	240.00	132.65	681.30
2	41.4%	88.5%	11.0%	100.0%	27.0%	73.7%	15.8%	100.0%	27.90	276.73	134.02	785.80
3	57.1%	93.9%	8.3%	100.0%	42.0%	85.8%	11.8%	100.0%	41.80	332.79	151.19	766.70
4	39.2%	83.0%	11.1%	100.0%	25.0%	62.1%	12.7%	99.0%	18.30	242.32	123.94	756.20
5	21.9%	74.3%	17.3%	100.0%	20.0%	61.9%	17.6%	100.0%	8.50	86.18	54.14	420.10
6	52.9%	89.3%	10.1%	100.0%	29.0%	73.9%	15.4%	100.0%	49.70	278.73	117.95	687.40
7	64.2%	94.1%	6.5%	100.0%	29.0%	80.8%	13.4%	100.0%	122.00	476.59	157.74	853.90
8	25.0%	74.6%	16.5%	100.0%	30.0%	73.6%	15.7%	100.0%	19.00	220.85	122.12	706.20
9	43.5%	82.0%	13.8%	100.0%	26.0%	67.1%	17.0%	100.0%	24.50	151.96	77.28	529.90
10	59.2%	91.9%	8.3%	100.0%	37.0%	77.6%	14.5%	100.0%	32.60	313.46	160.06	761.70
11	62.8%	94.6%	6.1%	100.0%	41.0%	82.3%	13.1%	100.0%	166.60	524.96	160.80	864.60
12	71.6%	95.9%	1.9%	100.0%	42.0%	84.7%	12.3%	100.0%	213.50	602.87	158.66	914.30
13	70.0%	96.6%	0.6%	100.0%	40.0%	86.3%	11.8%	100.0%	252.40	641.11	150.23	941.70
14	75.0%	97.2%	1.0%	100.0%	46.0%	87.9%	10.9%	100.0%	315.90	668.75	141.73	941.80
15	35.4%	81.9%	7.3%	100.0%	24.0%	67.1%	17.0%	100.0%	25.20	149.58	79.13	503.90
16	56.4%	92.0%	1.3%	100.0%	34.0%	77.5%	14.6%	100.0%	83.90	370.46	139.83	765.00
17	68.6%	94.9%	2.5%	100.0%	39.0%	82.5%	13.0%	100.0%	169.00	525.89	161.51	865.50
18	37.0%	73.7%	9.0%	100.0%	22.0%	53.6%	11.1%	96.0%	22.10	114.70	51.88	381.00
19	55.7%	88.3%	2.7%	100.0%	33.0%	67.3%	12.3%	98.0%	76.80	344.16	127.39	788.00
20	68.6%	93.0%	1.3%	100.0%	37.0%	74.5%	12.8%	99.0%	190.10	514.61	135.00	837.00
21	70.8%	94.5%	0.6%	100.0%	37.0%	78.0%	12.6%	100.0%	251.70	601.41	123.61	895.50
22	66.3%	95.3%	0.7%	100.0%	45.0%	87.9%	10.3%	100.0%	77.00	414.32	158.84	777.70
23	70.2%	96.5%	1.0%	100.0%	51.0%	89.9%	9.1%	100.0%	163.20	547.55	187.41	874.80
24	73.4%	97.0%	0.6%	100.0%	52.0%	91.1%	8.4%	100.0%	184.80	610.08	189.21	937.50
25	74.6%	97.5%	3.9%	100.0%	55.0%	92.0%	7.9%	100.0%	248.80	644.67	177.48	948.10

Table 5.2: Mean Tardiness: short term estimation: Generated set characteristics.

#	NNet	Data set	size	METHOD 1: Tolerance approach					METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	ALPHA(70)	ALPHA(100)	mean	(Std)	mean	(Std)
1	E5_1_1	Training set	600	95.2%	78.5%	57.3%	6.2%	0.3%	21.8	17.6	35.9	18.7
		Test set	350	99.4%	94.9%	88.3%	39.7%	16.3%	40.5	36.9	67.7	39.2
2	E5_1_2	Training set	600	95.2%	77.3%	54.5%	6.0%	1.3%	20.5	18.5	35.7	20.9
		Test set	350	99.4%	95.4%	88.9%	44.3%	16.9%	40.4	35.5	69.7	38.3
3	E5_2_1	Training set	600	99.0%	89.2%	72.0%	15.7%	5.5%	28.3	24.0	46.7	26.0
		Test set	350	99.4%	97.7%	94.9%	71.1%	54.6%	78.8	70.8	124.2	75.6
4	E5_2_2	Training set	600	98.2%	92.5%	82.0%	27.8%	8.3%	34.6	28.7	56.7	30.9
		Test set	350	99.7%	98.9%	96.3%	76.9%	52.9%	77.9	64.1	122.2	69.2
5	E5_3_1	Training set	600	97.2%	88.0%	69.8%	14.3%	2.5%	26.4	21.6	44.3	23.5
		Test set	350	100.0%	97.1%	93.7%	64.3%	49.1%	70.9	61.4	108.4	66.7
6	E5_3_2	Training set	600	99.0%	93.3%	77.8%	21.5%	4.2%	31.2	24.9	51.5	26.2
		Test set	350	99.4%	99.1%	95.4%	71.7%	49.1%	70.3	57.2	109.4	59.8
7	E5_4_1	Training set	600	98.0%	88.8%	70.3%	13.5%	2.8%	27.0	22.5	44.5	23.8
		Test set	350	100.0%	98.9%	96.0%	75.4%	54.9%	80.8	70.0	125.7	75.5
8	E5_4_2	Training set	600	99.7%	96.8%	88.3%	35.3%	12.2%	39.1	31.0	63.6	31.9
		Test set	350	100.0%	99.1%	96.6%	76.6%	60.0%	85.5	72.2	134.1	76.3
9	E5_5_1	Training set	600	92.7%	62.2%	35.2%	1.8%	0.2%	17.0	14.5	27.5	15.9
		Test set	350	94.9%	79.1%	62.6%	26.3%	15.4%	38.1	40.9	54.9	45.6
10	E5_5_2	Training set	600	86.2%	47.5%	19.0%	0.0%	0.0%	12.7	10.4	21.1	10.9
		Test set	350	95.7%	80.0%	63.1%	23.7%	11.4%	34.8	34.6	50.7	38.7
11	E5_6_1	Training set	600	98.3%	90.3%	75.0%	17.0%	3.5%	29.0	23.6	47.1	24.4
		Test set	350	99.4%	98.6%	93.7%	69.4%	51.7%	73.6	64.6	116.4	72.4
12	E5_6_2	Training set	600	99.2%	93.3%	80.0%	23.3%	8.0%	32.5	27.4	54.7	29.2
		Test set	350	99.4%	98.9%	96.9%	77.1%	59.4%	78.3	67.1	127.0	71.8
13	E5_7_1	Training set	600	99.7%	93.0%	81.5%	20.7%	6.3%	32.1	26.2	53.4	27.2
		Test set	350	100.0%	99.1%	96.0%	69.1%	46.3%	67.6	62.9	113.4	73.6
14	E5_7_2	Training set	600	98.8%	95.0%	84.5%	29.5%	10.0%	35.0	28.4	58.1	29.3
		Test set	350	100.0%	98.9%	97.4%	75.7%	56.0%	73.6	66.4	127.1	74.7
15	E5_8_1	Training set	600	97.8%	88.3%	71.8%	13.0%	2.2%	26.5	22.0	44.0	22.8
		Test set	350	99.7%	95.7%	91.7%	67.4%	45.1%	67.9	60.2	105.0	64.9
16	E5_8_2	Training set	600	99.5%	89.8%	76.7%	17.3%	4.2%	29.9	24.5	49.7	25.4
		Test set	350	99.7%	98.6%	95.4%	70.9%	51.4%	71.7	59.7	112.6	61.4

Table 5.3: Mean Tardiness: short term estimation: Results of metamodels

#	NNet	Data set	METHOD 4: deviation -to- flow time approach						TARDINESS			
			E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
1	E5_1_1	Training set	6%	5%	9%	8%	6%	6%	3.9	237.6	134.6	622.4
		Test set	13%	13%	15%	13%	11%	12%	5.9	244.1	129.2	681.3
2	E5_1_2	Training set	6%	6%	8%	7%	5%	5%	3.9	237.6	134.6	622.4
		Test set	15%	14%	16%	19%	11%	12%	5.9	244.1	129.2	681.3
3	E5_2_1	Training set	8%	6%	9%	7%	7%	6%	27.9	275.2	132.5	722.5
		Test set	20%	18%	25%	21%	22%	21%	29.7	279.3	136.6	785.8
4	E5_2_2	Training set	9%	8%	12%	10%	10%	9%	27.9	275.2	132.5	722.5
		Test set	20%	17%	27%	21%	23%	19%	29.7	279.3	136.6	785.8
5	E5_3_1	Training set	6%	6%	10%	9%	7%	6%	41.8	330.6	150.2	766.7
		Test set	17%	15%	20%	16%	16%	17%	46.9	336.6	153.0	748.2
6	E5_3_2	Training set	7%	6%	12%	10%	9%	7%	41.8	330.6	150.2	766.7
		Test set	16%	14%	20%	17%	16%	14%	46.9	336.6	153.0	748.2
7	E5_4_1	Training set	8%	6%	10%	8%	8%	7%	18.3	239.3	120.8	756.2
		Test set	22%	17%	27%	25%	27%	27%	22.8	247.5	129.0	657.4
8	E5_4_2	Training set	11%	9%	15%	13%	13%	12%	18.3	239.3	120.8	756.2
		Test set	23%	20%	31%	27%	28%	27%	22.8	247.5	129.0	657.4
9	E5_5_1	Training set	12%	9%	11%	9%	7%	6%	8.5	87.0	55.4	420.1
		Test set	25%	25%	23%	22%	16%	16%	11.8	84.8	52.0	298.2
10	E5_5_2	Training set	7%	6%	8%	6%	5%	4%	8.5	87.0	55.4	420.1
		Test set	21%	21%	21%	19%	14%	14%	11.8	84.8	52.0	298.2
11	E5_6_1	Training set	8%	7%	10%	8%	8%	7%	49.7	278.6	118.5	687.4
		Test set	17%	14%	24%	24%	21%	21%	57.0	279.0	117.1	678.4
12	E5_6_2	Training set	9%	8%	11%	10%	9%	9%	49.7	278.6	118.5	687.4
		Test set	19%	17%	22%	21%	19%	18%	57.0	279.0	117.1	678.4
13	E5_7_1	Training set	5%	5%	6%	6%	6%	5%	122.0	478.6	157.3	853.9
		Test set	10%	9%	12%	11%	11%	9%	129.6	473.1	158.6	843.5
14	E5_7_2	Training set	6%	5%	7%	6%	7%	6%	122.0	478.6	157.3	853.9
		Test set	11%	10%	12%	13%	11%	11%	129.6	473.1	158.6	843.5
15	E5_8_1	Training set	7%	6%	8%	7%	6%	6%	19.0	219.2	123.7	655.1
		Test set	17%	15%	19%	17%	15%	14%	24.2	223.7	119.3	706.2
16	E5_8_2	Training set	8%	6%	9%	8%	7%	6%	19.0	219.2	123.7	655.1
		Test set	18%	16%	23%	19%	17%	15%	24.2	223.7	119.3	706.2

Table 5.3 (Cont'd): Mean Tardiness: short term estimation: Results of metamodels

#	NNet	Data set	size	METHOD 1: Tolerance approach					METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	ALPHA(70)	ALPHA(100)	mean	(Std)	mean	(Std)
17	E5_9_1	Training set	600	97.8%	87.2%	73.8%	23.0%	7.7%	33.2	28.6	51.3	30.9
		Test set	350	98.3%	89.4%	76.3%	26.0%	11.4%	35.6	33.2	55.7	36.9
18	E5_9_2	Training set	600	98.5%	88.5%	75.2%	22.5%	7.8%	33.9	28.6	51.9	31.0
		Test set	350	98.0%	88.3%	77.4%	27.7%	10.9%	35.6	32.6	55.9	36.3
19	E5_10_1	Training set	600	99.8%	97.2%	91.5%	51.0%	28.3%	50.0	41.3	80.3	44.4
		Test set	350	99.1%	97.4%	92.9%	55.7%	32.3%	51.3	41.4	84.5	43.2
20	E5_10_2	Training set	600	99.7%	98.3%	94.3%	58.8%	31.3%	53.3	42.0	85.5	43.2
		Test set	350	99.1%	97.4%	93.7%	60.9%	32.6%	53.7	41.4	87.7	42.4
21	E5_11_1	Training set	600	99.8%	95.8%	88.3%	47.3%	24.8%	44.2	38.3	75.0	42.5
		Test set	350	100.0%	96.3%	90.0%	47.7%	24.3%	46.4	41.1	78.4	47.0
22	E5_11_2	Training set	600	99.5%	97.2%	89.8%	49.5%	27.2%	45.6	39.9	78.0	44.6
		Test set	350	99.4%	96.0%	89.7%	52.0%	26.6%	47.9	42.3	81.9	47.9
23	E5_12_1	Training set	600	99.5%	96.2%	88.3%	40.3%	16.5%	40.7	33.4	67.6	34.8
		Test set	350	100.0%	98.3%	93.1%	44.3%	19.1%	42.0	33.1	69.7	32.2
24	E5_12_2	Training set	600	99.7%	96.2%	87.0%	40.7%	17.2%	41.0	33.7	67.9	35.3
		Test set	350	100.0%	97.4%	90.9%	46.0%	19.4%	41.8	33.4	69.8	33.1
25	E5_13_1	Training set	600	99.5%	98.3%	95.0%	62.7%	43.8%	73.4	66.4	112.3	70.8
		Test set	350	100.0%	99.4%	95.7%	66.3%	45.4%	74.5	64.3	113.8	68.8
26	E5_13_2	Training set	600	99.7%	96.8%	91.0%	41.7%	14.8%	41.0	32.1	68.6	33.1
		Test set	350	100.0%	98.9%	93.7%	46.6%	17.4%	42.8	33.8	71.3	35.3
27	E5_14_1	Training set	600	99.3%	97.3%	91.8%	42.5%	16.5%	41.1	32.7	69.4	33.8
		Test set	350	99.1%	97.1%	91.4%	43.4%	17.7%	41.9	32.5	70.6	32.7
28	E5_14_2	Training set	600	99.2%	98.2%	91.0%	42.7%	17.5%	41.2	32.7	69.6	33.9
		Test set	350	99.1%	96.9%	91.1%	44.0%	18.6%	42.4	32.8	71.2	33.0
29	E5_15_1	Training set	600	92.7%	64.5%	37.8%	1.7%	0.0%	16.9	14.0	28.1	15.3
		Test set	350	99.7%	95.7%	88.0%	44.0%	24.0%	49.7	45.1	76.2	49.3
30	E5_15_2	Training set	600	95.5%	77.5%	56.2%	8.7%	1.5%	22.4	19.8	37.1	22.0
		Test set	350	98.9%	95.1%	87.4%	53.1%	34.0%	61.5	56.3	89.3	60.8
31	E5_16_1	Training set	600	99.2%	95.2%	86.7%	34.3%	12.8%	36.7	30.1	62.4	30.7
		Test set	350	100.0%	98.9%	96.0%	78.3%	58.9%	78.6	64.4	126.7	68.9
32	E5_16_2	Training set	600	99.3%	94.5%	87.5%	37.2%	13.2%	39.0	31.7	64.7	33.0
		Test set	350	99.7%	98.9%	98.3%	78.6%	62.6%	83.8	71.0	134.3	75.1

Table 5.3 (Cont'd): Mean Tardiness: short term estimation: Results of metamodels

#	NNet	Data set	METHOD 4: deviation -to- flow						TARDINESS					
			E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	time	appro	ach	min	mean	(Std)
17	E5_9_1	Training set	13%	11%	15%	12%	11%	10%	28.3	151.7	78.3	518.9		
		Test set	14%	11%	16%	14%	12%	11%	24.5	152.3	75.5	529.9		
18	E5_9_2	Training set	13%	11%	16%	12%	12%	10%	28.3	151.7	78.3	518.9		
		Test set	14%	11%	17%	14%	12%	11%	24.5	152.3	75.5	529.9		
19	E5_10_1	Training set	10%	9%	12%	11%	11%	10%	100.0	371.2	139.6	761.7		
		Test set	11%	9%	13%	11%	11%	9%	85.8	374.7	142.8	707.5		
20	E5_10_2	Training set	12%	10%	14%	12%	12%	10%	100.0	371.2	139.6	761.7		
		Test set	12%	11%	14%	12%	12%	9%	85.8	374.7	142.8	707.5		
21	E5_11_1	Training set	6%	5%	7%	7%	7%	6%	175.4	524.5	161.1	864.6		
		Test set	6%	6%	8%	7%	7%	5%	166.6	525.8	160.3	864.1		
22	E5_11_2	Training set	6%	5%	7%	7%	7%	6%	175.4	524.5	161.1	864.6		
		Test set	7%	6%	8%	7%	7%	5%	166.6	525.8	160.3	864.1		
23	E5_12_1	Training set	5%	4%	7%	6%	6%	5%	219.6	604.2	158.4	914.3		
		Test set	5%	4%	7%	6%	6%	5%	213.5	600.6	159.1	905.7		
24	E5_12_2	Training set	5%	4%	7%	6%	6%	5%	219.6	604.2	158.4	914.3		
		Test set	5%	4%	7%	6%	6%	5%	213.5	600.6	159.1	905.7		
25	E5_13_1	Training set	11%	15%	17%	18%	13%	13%	273.0	641.0	150.6	925.5		
		Test set	11%	13%	17%	19%	13%	13%	252.4	641.4	149.7	941.7		
26	E5_13_2	Training set	5%	4%	7%	6%	6%	5%	273.0	641.0	150.6	925.5		
		Test set	5%	4%	8%	7%	7%	5%	252.4	641.4	149.7	941.7		
27	E5_14_1	Training set	5%	4%	7%	6%	6%	5%	315.9	668.2	142.6	941.8		
		Test set	5%	4%	7%	5%	6%	5%	318.8	669.7	140.4	938.4		
28	E5_14_2	Training set	5%	4%	7%	6%	6%	5%	315.9	668.2	142.6	941.8		
		Test set	5%	4%	7%	5%	6%	5%	318.8	669.7	140.4	938.4		
29	E5_15_1	Training set	7%	6%	9%	7%	6%	5%	27.3	149.1	79.7	474.9		
		Test set	20%	17%	24%	23%	17%	16%	28.6	153.5	79.2	503.9		
30	E5_15_2	Training set	9%	7%	11%	10%	7%	7%	27.3	149.1	79.7	474.9		
		Test set	27%	25%	31%	32%	22%	20%	28.2	153.6	80.8	503.9		
31	E5_16_1	Training set	8%	7%	9%	8%	8%	7%	83.9	368.9	140.3	765.0		
		Test set	15%	16%	21%	18%	19%	17%	97.5	373.2	139.1	749.0		
32	E5_16_2	Training set	10%	8%	10%	9%	9%	7%	83.9	368.9	140.3	765.0		
		Test set	18%	19%	21%	19%	18%	16%	97.5	373.2	139.1	749.0		

Table 5.3 (Cont'd): Mean Tardiness: short term estimation: Results of metamodels

#	NNet	Data set	size	METHOD 1: Tolerance approach					METHOD 2: MAD		METHOD 3: MMAD	
				ALPHA(10)	ALPHA(20)	ALPHA(30)	ALPHA(70)	ALPHA(100)	mean	(Std)	mean	(Std)
33	E5_17_1	Training set	600	99.5%	93.8%	80.7%	20.7%	5.7%	32.1	25.5	53.1	26.0
		Test set	350	99.7%	98.3%	95.1%	69.7%	43.7%	64.8	59.6	110.9	70.9
34	E5_17_2	Training set	600	99.3%	96.2%	86.2%	28.3%	6.8%	34.6	26.6	58.1	26.7
		Test set	350	99.7%	99.4%	96.3%	76.9%	60.6%	73.3	62.7	125.2	68.2
35	E5_18_1	Training set	600	98.3%	85.2%	64.7%	14.0%	5.7%	28.2	25.4	44.9	29.0
		Test set	350	96.9%	85.1%	65.1%	14.9%	5.4%	28.3	25.9	45.1	30.0
36	E5_18_2	Training set	600	98.2%	86.8%	65.3%	16.8%	5.5%	28.6	26.2	46.5	29.7
		Test set	350	97.4%	84.0%	66.9%	15.7%	6.6%	28.9	26.8	46.7	31.4
37	E5_19_1	Training set	600	100.0%	98.8%	96.5%	66.5%	40.7%	59.4	44.8	94.6	43.7
		Test set	350	99.1%	98.0%	96.0%	70.9%	43.7%	60.4	46.3	97.6	46.2
38	E5_19_2	Training set	600	99.8%	99.0%	95.7%	68.7%	42.3%	60.2	46.0	96.5	45.1
		Test set	350	100.0%	99.1%	96.0%	72.8%	45.1%	61.7	46.5	100.3	45.3
39	E5_20_1	Training set	600	100.0%	98.7%	94.8%	58.2%	30.7%	51.5	41.4	86.2	43.8
		Test set	350	99.7%	97.4%	94.6%	62.0%	32.3%	54.3	48.8	91.3	53.0
40	E5_20_2	Training set	600	99.8%	98.0%	93.7%	59.0%	31.0%	51.5	41.4	86.1	43.8
		Test set	350	99.7%	99.4%	95.1%	61.7%	33.7%	54.9	48.3	91.1	52.1
41	E5_21_1	Training set	600	99.7%	99.2%	93.3%	48.0%	22.0%	44.8	35.9	75.2	35.8
		Test set	350	100.0%	98.9%	94.9%	51.1%	23.4%	47.8	38.9	79.1	39.8
42	E5_21_2	Training set	600	99.8%	97.8%	94.0%	50.3%	22.7%	45.3	36.0	75.8	35.8
		Test set	350	100.0%	98.6%	95.1%	55.4%	27.4%	48.6	38.6	80.6	38.8
43	E5_22_1	Training set	600	99.8%	98.2%	93.3%	65.5%	53.3%	82.3	69.1	120.4	73.1
		Test set	350	99.7%	97.1%	91.4%	64.6%	52.6%	82.3	66.3	116.2	70.4
44	E5_22_2	Training set	600	99.7%	97.2%	90.7%	45.8%	19.5%	43.9	35.4	72.1	36.1
		Test set	350	99.7%	96.0%	90.0%	44.0%	20.0%	45.3	36.7	71.7	39.2
45	E5_23_1	Training set	600	99.7%	96.8%	87.5%	31.7%	11.3%	36.7	29.3	61.7	30.4
		Test set	350	99.4%	96.3%	90.3%	33.1%	10.3%	37.5	28.8	62.5	28.4
46	E5_23_2	Training set	600	99.7%	96.8%	88.7%	33.8%	11.3%	37.0	29.5	62.5	30.3
		Test set	350	99.4%	96.0%	90.0%	36.0%	9.7%	37.5	29.1	62.8	29.1
47	E5_24_1	Training set	600	99.5%	98.0%	93.5%	60.3%	42.0%	95.5	87.9	128.7	92.4
		Test set	350	99.7%	98.9%	94.0%	61.1%	45.7%	96.7	87.7	130.3	92.8
48	E5_24_2	Training set	350	100.0%	97.7%	91.1%	37.7%	11.7%	37.6	30.0	64.8	29.4
		Test set	350	98.9%	96.6%	90.9%	40.9%	15.4%	39.5	32.5	66.8	32.9
49	E5_25_1	Training set	600	99.8%	97.5%	91.7%	59.0%	43.2%	85.3	77.3	117.4	81.6
		Test set	350	99.4%	96.6%	90.0%	62.6%	45.1%	85.9	75.1	116.9	77.8
50	E5_25_2	Training set	600	99.7%	96.3%	88.5%	37.7%	14.3%	38.7	30.3	64.3	30.6
		Test set	350	98.0%	94.9%	88.3%	41.7%	14.0%	39.8	32.1	66.7	33.6

Table 5.3 (Cont'd): Mean Tardiness: short term estimation: Results of metamodels

#	NNet	Data set	METHOD 4: deviation -to- flow time approach						TARDINESS			
			E4 - 1	(Std)	E4 - 2	(Std)	E4 - 3	(Std)	min	mean	(Std)	max
33	E5_17_1	Training set	5%	4%	7%	6%	8%	5%	169.0	525.9	162.9	865.5
		Test set	8%	6%	11%	12%	10%	9%	185.3	525.9	159.2	855.3
34	E5_17_2	Training set	4%	4%	7%	6%	6%	5%	169.0	525.9	162.9	865.5
		Test set	9%	9%	11%	10%	10%	9%	185.3	525.9	159.2	855.3
35	E5_18_1	Training set	13%	11%	17%	13%	13%	12%	24.1	114.0	51.8	381.0
		Test set	13%	10%	18%	13%	13%	14%	22.1	115.8	52.1	342.3
36	E5_18_2	Training set	14%	11%	16%	12%	13%	12%	24.1	114.0	51.8	381.0
		Test set	14%	11%	16%	12%	13%	15%	22.1	115.8	52.1	342.3
37	E5_19_1	Training set	13%	10%	16%	14%	14%	12%	76.8	342.5	127.8	788.0
		Test set	13%	9%	17%	14%	15%	13%	92.3	347.0	126.7	701.3
38	E5_19_2	Training set	13%	11%	17%	15%	15%	12%	76.8	342.5	127.8	788.0
		Test set	13%	10%	18%	15%	15%	13%	92.3	347.0	126.7	701.3
39	E5_20_1	Training set	7%	8%	9%	8%	8%	6%	191.7	514.9	135.1	837.0
		Test set	8%	6%	12%	52%	10%	32%	0.0	513.4	136.4	832.4
40	E5_20_2	Training set	7%	7%	9%	8%	8%	6%	191.7	514.9	135.1	837.0
		Test set	8%	7%	12%	51%	10%	32%	0.0	513.4	136.4	832.4
41	E5_21_1	Training set	5%	4%	8%	6%	7%	5%	251.7	601.4	123.2	890.5
		Test set	6%	5%	8%	7%	7%	5%	255.6	601.4	124.5	895.5
42	E5_21_2	Training set	6%	4%	8%	6%	7%	5%	251.7	601.4	123.2	890.5
		Test set	6%	5%	8%	7%	7%	5%	255.6	601.4	124.5	895.5
43	E5_22_1	Training set	16%	10%	17%	11%	16%	12%	77.0	413.7	159.4	777.7
		Test set	16%	9%	17%	10%	15%	11%	109.0	415.4	158.0	767.3
44	E5_22_2	Training set	7%	7%	12%	11%	9%	7%	77.0	413.7	159.4	777.7
		Test set	8%	7%	12%	11%	9%	8%	109.0	415.4	158.0	767.3
45	E5_23_1	Training set	5%	4%	7%	6%	6%	4%	163.2	545.8	186.7	874.8
		Test set	5%	4%	7%	5%	6%	4%	185.5	550.5	188.7	856.7
46	E5_23_2	Training set	5%	4%	7%	6%	6%	4%	163.2	545.8	186.7	874.8
		Test set	5%	4%	7%	5%	6%	5%	185.5	550.5	188.7	856.7
47	E5_24_1	Training set	18%	22%	24%	29%	20%	21%	206.0	608.3	187.9	937.5
		Test set	18%	21%	25%	30%	20%	22%	184.8	613.1	191.5	925.2
48	E5_24_2	Training set	5%	4%	7%	6%	6%	4%	234.0	605.2	188.3	937.5
		Test set	5%	5%	7%	7%	6%	5%	184.8	613.1	191.5	925.2
49	E5_25_1	Training set	15%	18%	19%	20%	15%	16%	248.8	643.2	176.6	932.1
		Test set	14%	16%	19%	20%	16%	16%	271.8	647.2	179.0	948.1
50	E5_25_2	Training set	4%	4%	7%	5%	6%	5%	248.8	643.2	176.6	932.1
		Test set	4%	4%	7%	5%	6%	5%	271.8	647.2	179.0	948.1

Table 5.3 (Cont'd): Mean Tardiness: short term estimation: Results of metamodels

Appendix C

Codes

Sheet 1:
System Simulation (SIMAN) Model Frame

```

BEGIN;
  create;

  read:IN,,1:LAMDA1,LAMDA2,LAMDA3,LAMDA4,LAMDA5,LAMDA6,RULE,k;
  assign:poptime=0;
  duplicate:1,r2:1,1,1,r5:1,r6;

r1  delay:LAMDA1;
    duplicate:1,r1;
  assign:TYPE=1;N5=1;M1=ENTRANCE;
        OPTIME1=0;
        OPTIME2=4;
        OPTIME3=5;
        OPTIME4=9;
        OPTIME5=0;
        OPTIME6=0;
        OPTIME7=0;
    toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
        OPTIME7;mark(ARRT);
  assign:DD=tnow+(1.0*toptime);
  count:1,1;next(tran0);

r2  delay:LAMDA2;
    duplicate:1,r2;
  assign:TYPE=2;N5=1;M1=ENTRANCE;
        OPTIME1=0;
        OPTIME2=6;
        OPTIME3=6;
        OPTIME4=9;
        OPTIME5=0;
        OPTIME6=0;
        OPTIME7=0;
    toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
        OPTIME7;mark(ARRT);
  assign:DD=tnow+(1.0*toptime);
  count:2,1;next(tran0);

r3  delay:lamda3;
    duplicate:1,r3;
  assign:TYPE=3;N5=1;M1=ENTRANCE;
        OPTIME1=9;
        OPTIME2=11;
        OPTIME3=0;
        OPTIME4=0;
        OPTIME5=3;
        OPTIME6=0;
        OPTIME7=0;
    toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
        OPTIME7;mark(ARRT);

```

Sheet 1:(Cont'd)

```

assign:DD=(now+L)*toptime);
count:3,I:next@trans);

r4  delay:LAMDA4;
duplicate:I,r4;
assign:TYPE=4;N=4;I:ENTRANCE:
OPTIME1=1;
OPTIME2=1;
OPTIME3=1;
OPTIME4=1;
OPTIME5=1;
OPTIME6=1;
OPTIME7=1;
toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
OPTIME7;mod(CARRT);
assign:DD=(now+L)*toptime);
count:4,I:next@trans);

r5  delay:LAMDA5;
duplicate:I,r5;
assign:TYPE=5;N=5;I:ENTRANCE:
OPTIME1=1;
OPTIME2=1;
OPTIME3=1;
OPTIME4=1;
OPTIME5=1;
OPTIME6=1;
OPTIME7=1;
toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
OPTIME7;mod(CARRT);
assign:DD=(now+L)*toptime);
count:5,I:next@trans);

r6  delay:LAMDA6;
duplicate:I,r6;
assign:TYPE=6;N=6;I:ENTRANCE:
OPTIME1=1;
OPTIME2=1;
OPTIME3=1;
OPTIME4=1;
OPTIME5=1;
OPTIME6=1;
OPTIME7=1;
toptime=OPTIME1+OPTIME2+OPTIME3+OPTIME4+OPTIME5+OPTIME6+
OPTIME7;mod(CARRT);
assign:DD=(now+L)*toptime);
count:6,I:next@trans);

trans queue,AGV,Q;
request:AGV,CDS,Q;
transport:AGV,SE,Q;

```

Sheet 1:(Cont'd)

```

station,2-8:

free:AGV;
  assign:od=0;
  assign:poptime=poptime+optime;
  assign:od=arrt+0*(delarrt/toptime)*poptime);
  assign:MODD=max(od,tnow+optime);
branch,1:if, rule==1,p1;
  if, rule==2,p;
  if, rule==3,p;
p1  assign:priority=optime*next(q);
p2  assign:priority=ld:next(q);
p3  assign:priority=MODD*next(q);
l3  assign:MODD=max(od,tnow+optime):temp1=m;
  queue,M+7;
  wait:temp1;
q   queue,M;
  seize:M-1;
  delay:OPTIME.;
  assign:temp=next(m)-1:next(l4);
l2  remove:1:m,l3;
l4  assign:temp=temp-1;
  branch,1;
  if, temp==0,l1;
  else l2;
l1  signal:m;
  delay:0.00001;
  release:M-1:next(0:m);

station,STATF@S@ENTF:
free:AGV;
tally:type,int(A,RRT);
count:type+6,l;
tally:type+6,mx(0,tnow-DD);
  branch,1:if, tnow < dd,l1;
  if, tnow > dd,l2;
t1  count:type+12,l:dispose;
t2  count:type+13,l:dispose;

create,1,9200;

;write to output file:
;average type 1,2,3,4,5,6 job tardiness
;average type 1,2,3,4,5,6 job flow time
;machine util M1 to M7
;counter for number of type 1,2,3,4,5,6 jobs tardy
;counter for total number of type 1,2,3,4,5,6 jobs processed

write,OUT "f8.1,f8.1,f8.1,f8.1,f8.1,f8.1,f8.1,f8.1,f8.1,
f8.1,f8.1,f8.1,f8.1,f8.3,f8.3,f8.3,f8.3,
f8.3,f8.3,f8.3,f8.1,f8.1,f8.1,f8.1,f8.1,

```


18.1,18.1,18.1,18.1,18.1,18.1)",1:

Sheet 1:(Cont'd)

tavg(tardiness of job 1),tavg(tardiness of job 2),
 tavg(tardiness of job 3),tavg(tardiness of job 4),
 tavg(tardiness of job 5),tavg(tardiness of job 6),
 tavg(FLOW TIME OF JOB 1),tavg(FLOW TIME OF JOB 2),
 tavg(FLOW TIME OF JOB 3),tavg(FLOW TIME OF JOB 4),
 tavg(FLOW TIME OF JOB 5),tavg(FLOW TIME OF JOB 6),
 DAVG(M1 UTIL U1),DAVG(M2 UTIL U2),DAVG(M3 UTIL U3),
 DAVG(M4 UTIL U4),DAVG(M5 UTIL U5),DAVG(M6 UTIL U6),
 DAVG(M7 UTIL U7),
 nc(# of type 1 jobs tardy),nc(# of type 2 jobs tardy),
 nc(# of type 3 jobs tardy),nc(# of type 4 jobs tardy),
 nc(# of type 5 jobs tardy),nc(# of type 6 jobs tardy),
 nc(# of type 1 jobs processed),nc(# of type 2 jobs processed),
 nc(# of type 3 jobs processed),nc(# of type 4 jobs processed),
 nc(# of type 5 jobs processed),nc(# of type 6 jobs processed),:dispose;

Sheet 2:
System Simulation - SIMAN Experimental Frame

```

begin;
project.exp # 4 Large System case, SOUHEYL TOUHAMI;

attributes:optime1:optime2:optime3:optime4:
           optime5:optime6:optime7:
           TYPE:ARRIVE:OPTIME:POPTIME:DD:
           OD:MODD:temp1
           LAMDA1:LAMDA2:LAMDA3:LAMDA4:LAMDA5:LAMDA6:priority:rule;

variables:temp;

stations:ENTRANCE:s1:s2:s3:s4:s5
         :s6:s7:STATIONEXIT;

queues:1,AGVQ:
        2,Q1.LVF(PRIORITY);
        3,Q2.LVF(PRIORITY);
        4,Q3.LVF(PRIORITY);
        5,Q4.LVF(PRIORITY);
        6,Q5.LVF(PRIORITY);
        7,Q6.LVF(PRIORITY);
        8,Q7.LVF(PRIORITY);
        9,Q21.LVF(PRIORITY);
        10,Q22.LVF(PRIORITY);
        11,Q23.LVF(PRIORITY);
        12,Q24.LVF(PRIORITY);
        13,Q25.LVF(PRIORITY);
        14,Q26.LVF(PRIORITY);
        15,Q27.LVF(PRIORITY);

resources:MACHINE1:MACHINE2:MACHINE3:MACHINE4:
          MACHINE5:MACHINE6:MACHINE7;

sequences:1,s1 , OPTIME = optime1 &
          s3 , OPTIME = optime2 &
          s2 , OPTIME = optime3 &
          s4 , OPTIME = optime1 &
          STATIONEXIT;
2,s1 , OPTIME = optime1 &
  s2 , OPTIME = optime1 &
  s3 , OPTIME = optime3 &
  STATIONEXIT;
3,s3 , OPTIME = optime1 &
  s4 , OPTIME = optime1 &
  s2 , OPTIME = optime3 &
  STATIONEXIT;

```

Sheet 2:(Cont'd)

4,,s6 . OPTIME = optime1 &
 s2 . OPTIME = optime1 &
 s4 . OPTIME = optime3 &
 s7 . OPTIME = optime4 &
 s1 . OPTIME = optime5 &
 STATIONEXIT;
 5,,s3 . OPTIME = optime1 &
 s5 . OPTIME = optime2 &
 s1 . OPTIME = optime3 &
 s7 . OPTIME = optime4 &
 s2 . OPTIME = optime5 &
 s6 . OPTIME = optime6 &
 STATIONEXIT;
 6,,s2 . OPTIME = optime1 &
 s5 . OPTIME = optime1 &
 s1 . OPTIME = optime3 &
 s7 . OPTIME = optime4 &
 s6 . OPTIME = optime5 &
 s3 . OPTIME = optime6 &
 s4 . OPTIME = optime7 &
 STATIONEXIT;

transporters:1,AGV,3,1,10;

distances:1,1-9,20,10,17,10,20,19,15,16/
 10,6,15,17,14,10,17/
 5,5,7,9,15,12/
 10,9,11,18,10/
 9,11,7,13/
 10,6,15/
 5,9/
 10;

tallies:1,FLOW TIME OF JOB 1;
 2,FLOW TIME OF JOB 2;
 3,FLOW TIME OF JOB 3;
 4,FLOW TIME OF JOB 4;
 5,FLOW TIME OF JOB 5;
 6,FLOW TIME OF JOB 6;
 7,TARDINESS OF JOB 1;
 8,TARDINESS OF JOB 2;
 9,TARDINESS OF JOB 3;
 10,TARDINESS OF JOB 4;
 11,TARDINESS OF JOB 5;
 12,TARDINESS OF JOB 6;

counters:1,# OF TYPE 1 JOBS ARRIVED;
 2,# OF TYPE 2 JOBS ARRIVED;
 3,# OF TYPE 3 JOBS ARRIVED;
 4,# OF TYPE 4 JOBS ARRIVED;
 5,# OF TYPE 5 JOBS ARRIVED;
 6,# OF TYPE 6 JOBS ARRIVED;

Sheet 2:(Cont'd)

7,# OF TYPE 1 JOBS PROCESSED:
 8,# OF TYPE 2 JOBS PROCESSED:
 9,# OF TYPE 3 JOBS PROCESSED:
 10,# OF TYPE 4 JOBS PROCESSED:
 11.# OF TYPE 5 JOBS PROCESSED:
 12.# OF TYPE 6 JOBS PROCESSED:
 13.# OF TYPE 1 JOBS ontime:
 14.# OF TYPE 2 JOBS ontime:
 15.# OF TYPE 3 JOBS ontime:
 16.# OF TYPE 4 JOBS ontime:
 17.# OF TYPE 5 JOBS ontime:
 18.# OF TYPE 6 JOBS ontime:
 19.# OF TYPE 1 JOBS tardy:
 20.# OF TYPE 2 JOBS tardy:
 21.# OF TYPE 3 JOBS tardy:
 22.# OF TYPE 4 JOBS tardy:
 23.# OF TYPE 5 JOBS tardy:
 24.# OF TYPE 6 JOBS tardy;

dstats:NQ(AGVQ).WIP for AGVs:
 NT(AGV)/3.UTILIZATION of AGVs:
 NQ(Q1).WIP for Q1:
 NQ(Q2).WIP for Q2:
 NQ(Q3).WIP for Q3:
 NQ(Q4).WIP for Q4:
 NQ(Q5).WIP for Q5:
 NQ(Q6).WIP for Q6:
 NQ(Q7).WIP for Q7:
 NR(MACHINE1).M1 UTIL U1:
 NR(MACHINE2).M2 UTIL U2:
 NR(MACHINE3).M3 UTIL U3:
 NR(MACHINE4).M4 UTIL U4:
 NR(MACHINE5).M5 UTIL U5:
 NR(MACHINE6).M6 UTIL U6:
 NR(MACHINE7).M7 UTIL U7;

files:1,IN,"t4_in.mat".SEQ.FPBE:
 2,OUT,"t4_out.mat".SEQ.free;

replicate,1000,0,9200 ...1840;

Sheet 3:**Input Data Generator SIMAN Model Frame**

```

begin;

create,1000;

assign:LAMDA1=UNIF(0,20,100,1);
      LAMDA2=UNIF(0,100,1);
      LAMDA3=UNIF(0,100,3);
      LAMDA4=UNIF(0,100,1);
      LAMDA5=UNIF(0,100,5);
      LAMDA6=UNIF(0,100,6);
      RULE=DISC(1/2,2/3,2,1,3,7);
      k=DISC(1/8,2,2/3,1,3/8,1,1/8,5,5/8,6,6/8,7,7/8,8,1,9,8);

write,IN,FILE=LAMDA1,LAMDA2,LAMDA3,LAMDA4,LAMDA5,LAMDA6,RULE,k:dispose;

```

Input Data Generator SIMAN Experimental Frame

```

begin;

project.INPUT DATA CREATION, SOUHEYL TOUHAMI;

attributes:LAMDA1:
           LAMDA2:
           LAMDA3:
           LAMDA4:
           LAMDA5:
           LAMDA6:
           RULE:
           k:

seeds:1,567;2,1345;3,314,908;
      5,7485;6,3953;7,9418,756;8,9,4476;10,376;

files:1,IN,"t4_1in.mn",SEQ,FILE;

replicate;

```