

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

PhD. THESIS

Elrasheed ISMAIL MOHOMMOUD ZAYID

**PREDICTING PERFORMANCE MEASURES OF A MULTIPROCESSOR
ARCHITECTURE BY USING MACHINE LEARNING METHODS**

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA, 2012

**ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**PREDICTING PERFORMANCE MEASURES OF A MULTIPROCESSOR
ARCHITECTURE BY USING MACHINE LEARNING METHODS**

Elrasheed ISMAIL MOHOMMOUD ZAYID

Ph.D THESIS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the ***Philosophiae Doctor*** of Science by the board of jury on 31 / 12 / 2012.

.....
Asst. Prof. Dr. Mehmet Fatih AKAY
SUPERVISOR

.....
Assoc.Prof.Dr. Zekeriya TÜFEKÇİ
MEMBER

.....
Assoc.Prof.Dr. Mustafa GÖK
MEMBER

.....
Asst.Prof.Dr. Mustafa ORAL
MEMBER

.....
Asst.Prof.Dr. Serdar YILDIRIM
MEMBER

This Ph. D Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.

Registration Number :

**Prof. Dr. Selahattin SERİN
Director
Institute of Natural and Applied Sciences**

**This study was supported by Ç.Ü.Research Projects Unit
Project Number: MMF2011D8**

Note: The usage of the presented specific declarations, tables, figures, and photographs either in this thesis or in any other reference without citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic.

ABSTRACT

PhD. THESIS

PREDICTING PERFORMANCE MEASURES OF A MULTIPROCESSOR ARCHITECTURE BY USING MACHINE LEARNING METHODS

Elrasheed ISMAIL MOHOMMOUD ZAYID

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Supervisor : Asst. Prof. Dr. Mehmet Fatih AKAY

Year : 2012, Pages: 91

Jury : Asst. Prof. Dr. Mehmet Fatih AKAY
: Assoc.Prof.Dr. Mustafa GÖK
: Assoc.Prof.Dr. Zekeriya TÜFEKÇİ
: Asst.Prof.Dr. Mustafa ORAL
: Asst.Prof.Dr. Serdar YILDIRIM

In this thesis, we develop machine learning models for predicting the performance measures of both a message passing and a distributed shared memory multiprocessor architecture interconnected by the Simultaneous Optical Multiprocessor Exchange Bus (SOME-Bus), which is a fiber-optic interconnection network. Machine learning models include multi-layer feed-forward artificial neural networks (MFANN's), support vector regression (SVR) and generalized regression neural networks (GRNN). OPNET Modeler is used to simulate the SOME-Bus multiprocessor architecture and to create the training and testing datasets. The simulation has been run under different traffic patterns including uniform, hot-region, perfect shuffle and bit-reverse for varying values of the ratio of the average channel transfer time to the average thread run time (T/R). Client-server and asynchronous traffic models are considered for the message passing protocol. Using different number of cross validations, the performance of the machine learning prediction models are evaluated using standard error of estimate (SEE), multiple correlation coefficient (R), mean absolute error (MAE), relative absolute error (RAE) and root relative square error ($RRSE$). It is shown that MFANN models perform better (i.e., lower SEE , MAE , RAE , $RRSE$ and higher R) than GRNN-based, SVR-based and multiple linear regression (MLR) based models for predicting the performance measures of a message passing and distributed shared memory multiprocessor architecture.

Keywords: Multiprocessor architectures, message passing, distributed shared memory, artificial neural networks, support vector regression.

ÖZ

DOKTORA TEZİ

MAKİNE ÖĞRENMESİ METODLARI KULLANILARAK ÇOKLU İŞLEMCİ MİMARİSİNİN PERFORMANS ÖLÇÜMLERİNİ TAHMİN ETME

Elrasheed ISMAIL MOHOMMOUD ZAYID

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Asst. Prof. Dr. Mehmet Fatih AKAY

Year : 2012, Pages: 91

Jüri : Asst. Prof. Dr. Mehmet Fatih AKAY
: Assoc.Prof.Dr. Mustafa GÖK
: Assoc.Prof.Dr. Zekeriya TÜFEKÇİ
: Asst.Prof.Dr. Mustafa ORAL
: Asst.Prof.Dr. Serdar YILDIRIM

Bu çalışmada, çoklu işlemciye sahip mesaj geçişi ve dağıtık ortak hafıza mimarilerinin başarımlarını ölçütlerini tahmin eden makine öğrenmesi modelleri geliştirilmiştir. Mesaj geçişi ve dağıtık ortak hafıza mimarileri, ara bağlantı ağı olarak fiber-optik SOME-Bus'ı kullanılmaktadır. Çalışmada kullanılan makine öğrenmesi modelleri Çoklu-Katmanlı İleri-Beslemeli Yapay Sinir Ağları (ÇKİBYSA), Destek Vektör Makineleri (DVM) ve Genelleştirilmiş Regresyon Sinir Ağları (GRSA)'dır. SOME-Bus çoklu işlemci ağının benzetimi ve ayrıca eğitim ve test veri kümelerinin elde edilmesi için OPNET Modeller kullanılmıştır. Tasarlanan benzetim modelleri, normal dağılım, yoğun bölge, mükemmel karışım ve bit dönüşümü trafikleri altında ortalama kanal iletim zamanının (T) ortalama iplik çalışma zamanına oranı (R) olan (T/R) ifadesinin değişik değerleri için koşuturulmuştur. Sunucu-istemci ve asenkron trafik modelleri mesaj geçişi protokolünde kullanılmıştır. kullanılarak makine öğrenmesi tahmin modellerinin başarımlarını değişik sayıda çapraz doğrulama, standart tahmin hatası (STH), çoklu korelasyon katsayısı ($ÇKK$), ortalama mutlak hata (OMH), bağıl mutlak hata (BMH) ve kök bağıl karesel hata ($KBKH$) kullanılarak değerlendirilmiştir. Çalışmanın sonucunda Çoklu-Katmanlı İleri-Beslemeli Yapay Sinir Ağları ile tasarlanan model (düşük STH , OMH , BMH , $KBKH$ ve yüksek $ÇKK$), GRSA tabanlı, DVM tabanlı ve Çoklu Doğrusal Regresyon (ÇDR) tabanlı modele göre mesaj geçişi ve dağıtık ortak hafıza mimarilerinin başarımlarını ölçütlerini tahmin etmekte daha iyi sonuçlar üretmiştir.

Anahtar Kelimeler: Çoklu işlemcili mimariler, mesaj geçişi, dağıtık ortak hafıza, yapay sinir ağları, destek vektör regresyonu

ACKNOWLEDGMENTS

I am deeply grateful to my advisor *Asst.Prof.Dr. M. Fatih AKAY* for his guidance to accomplish this thesis. I really appreciate all his comments and suggestions. Special thanks to him for his help, support and patience over the years.

I would also like to thank Assoc.Prof.Dr. Mustafa GÖK, Assoc.Prof.Dr. Zekeriya TÜFEKÇİ, Asst.Prof.Dr. Mustafa ORAL and Asst.Prof.Dr. Serdar YILDRIM for serving in my committee.

I would like to express my gratitude to *The Ministry of Higher Education* and *The University of Elimam Elmahdi* in *Sudan*. I would like to express my appreciation to *The Turkish Government* and *The Ministry of Education* for offering me this opportunity, hosting me to accomplish this work and for their kind hospitality.

I would like to thank again *Assoc.Prof. Dr. Mustafa GÖK* and *Ali KARAMAN* for their honest friendship and brotherhood.

I am grateful to my colleagues *Erman AKTÜRK*, *Mustafa AÇIKKAR*, *Çiğdem ACI* and *İpek ABASIKELEŞ* for their help and cooperation.

I would like to thank *OPNET Technologies Inc.*, for letting me use the OPNET Modeler under the University Program and to Cukurova University Scientific Research Projects Center (Project no: MMF2011D8) for funding the thesis.

I would also like to thank *Dr. Constantine Katsinis* for letting me include the material about the SOME-Bus architecture in this thesis.

Finally, special thanks to my wife *Amel* and my *family* for their faith in me and their sacrifice, patience, encouragement and understanding.

CONTENTS	PAGE
ABSTRACT.....	I
ÖZ	II
ACKNOWLEDGMENTS	III
CONTENTS	IV
LIST OF TABLES.....	VI
LIST OF FIGURES.....	VIII
LIST OF ABBREVIATIONS.....	X
1. INTRODUCTION.....	1
1.1. Parallel Computing	1
1.2. Motivation and the Aim of the Thesis	3
1.3. Organization of the Thesis.....	5
1.4. Literature Review.....	5
2. OVERVIEW OF THE SOME-BUS ARCHITECTURE.....	11
2.1. The SOME-Bus Architecture	11
3. OVERVIEW OF METHODS.....	17
3.1. Multi-layer Feed-forward Artificial Neural Networks	17
3.2. Generalized Regression Neural Networks.....	18
3.3. Support Vector Regression.....	20
3.3.1. Linear Support Vector Regression.....	20
3.3.2. Non-linear Support Vector Regression	22
3.4. Multiple Linear Regression	23
4. SIMULATION AND DATASET GENERATION	25
4.1. Simulation Framework	25
4.2. MP Framework and Dataset Generation.....	25
4.3. DSM Framework and Dataset Generation.....	31
5. RESULTS AND DISCUSSION.....	39
5.1. MFANN Prediction Models	39
5.2. SVR Prediction Model.....	40
5.3. Performance Measures.....	42

5.4. Results and Discussion for MP with ACK's	43
5.5. Results and Discussion for MP without ACK's.....	45
5.6. Results and Discussion for Hybrid MP.....	47
5.7. Results and Discussion for DSM Results	48
6. CONCLUSION	79
REFERENCES.....	83
BIOGRAPHY	91

LIST OF TABLES	PAGE
Table 4.1. Synthetic Traffic Patterns.....	28
Table 4.2. Descriptive statistics of the MP with ACK's dataset	30
Table 4.3. Descriptive statistics of the MP without ACK's dataset	30
Table 4.4. Descriptive statistics of the Hybrid MP dataset.....	31
Table 4.5. System Parameters.....	35
Table 4.6. Descriptive statistics of the DSM dataset.....	36
Table 5.1. Performance measures of the MP with ACK using 10-fold CV.....	51
Table 5.2. Performance measures of the MP with ACK using 20-fold CV.....	51
Table 5.3. Performance measures of the MP with ACK using 30-fold CV.....	52
Table 5.4. Performance measures of the MP with ACK using 40-fold CV.....	52
Table 5.5. Performance measures of the MP with ACK using 50-fold CV.....	53
Table 5.6. Performance measures of the MP with ACK using 60-fold CV.....	53
Table 5.7. Performance measures of the MP with ACK using 70-fold CV.....	54
Table 5.8. Performance measures of the MP with ACK using 80-fold CV.....	54
Table 5.9. Performance measures of the MP without ACK using 10-fold CV	55
Table 5.10. Performance measures of the MP without ACK using 20-fold CV	55
Table 5.11. Performance measures of the MP without ACK using 30-fold CV	56
Table 5.12. Performance measures of the MP without ACK using 40-fold CV	56
Table 5.13. Performance measures of the MP without ACK using 50-fold CV	57
Table 5.14. Performance measures of the MP without ACK using 60-fold CV	57
Table 5.15. Performance measures of the MP without ACK using 70-fold CV	58
Table 5.16. Performance measures of the MP without ACK using 80-fold CV	58
Table 5.17. Performance measures of the Hybrid MP using 10-fold CV	59
Table 5.18. Performance measures of the Hybrid MP using 20-fold CV	59
Table 5.19. Performance measures of the Hybrid MP using 30-fold CV	60
Table 5.20. Performance measures of the Hybrid MP using 40-fold CV	60
Table 5.21. Performance measures of the Hybrid MP using 50-fold CV	61
Table 5.22. Performance measures of the Hybrid MP using 60-fold CV	61
Table 5.23. Performance measures of the Hybrid MP using 70-fold CV	62

Table 5.24. Performance measures of the Hybrid MP using 80-fold CV	62
Table 5.25. Performance measures of the DSM using 10-fold CV	63
Table 5.26. Performance measures of the DSM using 20-fold CV	63
Table 5.27. Performance measures of the DSM using 30-fold CV	64
Table 5.28. Performance measures of the DSM using 40-fold CV	64
Table 5.29. Performance measures of the DSM using 50-fold CV	65
Table 5.30. Performance measures of the DSM using 60-fold CV	65
Table 5.31. Performance measures of the DSM using 70-fold CV	66
Table 5.32. Performance measures of the DSM using 80-fold CV	66
Table 5.33. Training times for MFANN on MP-ACK's using different folds	67
Table 5.34. Training times for SVR-L on MP-ACK's using different folds	68
Table 5.35. Training times for SVR-RBF on MP-ACK's using different folds	69
Table 5.36. Training times for MFANN on MP-UNACK's using different folds	70
Table 5.37. Training times for SVR-L on MP-UNACK's using different folds	71
Table 5.38. Training times for SVR-RBF on MP-UNACK's using different folds ..	72
Table 5.39. Training times for MFANN on Hybrid MP using different folds	73
Table 5.40. Training times for SVR-L on Hybrid MP using different folds	74
Table 5.41. Training times for SVR-RBF on Hybrid MP using different folds	75
Table 5.42. Training times for MFANN on DSM using different folds	76
Table 5.43. Training times for SVR-L on DSM using different folds	77
Table 5.44. Training times for SVR-RBF on DSM using different folds	78

LIST OF FIGURES	PAGE
Figure 1.1. Shared-Memory v.s Distributed-Memory.....	2
Figure 2.1. Parallel Receiver Array	12
Figure 2.2. The SOME-Bus Optical Interface	14
Figure 2.3. The SOME-Bus Processor Interface.....	15
Figure 3.1. A Typical Multilayer Feed-Forward Neural Network	17
Figure 3.2. Architecture of Generalized Regression Neural Network Model	19
Figure 4.1. A typical N-node SOME-Bus Architecture Using MP Protocols	26
Figure 4.2. A Typical Process Model for the Queues	27
Figure 4.3. Node Model of a four-node DSM over SOME-Bus Architecture.....	32
Figure 5.1. MFANN Prediction Model	39

LIST OF ABBREVIATIONS

ACK's	: Acknowledgments
ANN	: Artificial neural network
A-Si	: Amorphous silicon
CMOS	: Complementary metal–oxide–semiconductor
CPU	: Central processing unit
CU	: Channel utilization
CV	: Cross validation
CWT	: Channel waiting time
DSM	: Distributed shared memory
GRNN	: Generalized regression neural network
IWT	: Input waiting time
L	: Linear
MAE	: Mean absolute error
MESI	: Modified exclusive shared invalid
MFANN	: Multilayer feed forward artificial neural network
MLR	: Multiple linear regression
MP	: Message passing
MSI	: Modified-shared invalid
NRT	: Network response time
NUMA	: Non-uniform memory access
P_{cf}	: Probability that the cache is full
P_m	: Probability that a block can be found in modified state
PU	: Average processor utilization
P_{uor}	: Probability of having an upgrade ownership request
P_w	: Probability that a data message is due to a write miss
R	: Multiple correlation coefficient
RAE	: Relative absolute error
RBF	: Radial basis function

RRSE : Root relative square error
SAS : Sharable address space
SEE : Standard error of estimation
SOME-Bus : Simultaneous optical multiprocessor exchange bus
SVM : Support vector machine
T/R : Ratio of the mean message channel transfer time to the mean thread
: run time

1. INTRODUCTION

1.1. Parallel Computing

High performance computing is required for many science-engineering domain applications. Some important domains for parallel computing nowadays include scientific applications that model physical phenomena; engineering applications such as those in computer-aided design, digital signal processing, automobile crash simulation and even simulations used to evaluate architectural tradeoffs; graphics and visualization applications that render scenes or volumes into images; optimization applications such as crew scheduling for an airline and transport control; artificial intelligence applications such as expert systems and robotics; multiprogrammed workloads; and the operating system itself, which is a particularly complex parallel application (Culler et al., 1999; Thiele et al., 2005 and Sendag et al., 2007).

Parallel computing is the simultaneous use of multiple compute processing units to solve a computational problem. Parallel computing takes hold in many areas of mainstream computing (Hennessy and Patterson, 2007). Developing parallel applications that are robust and provide good speed-up across current and future multiprocessors is a critical task and requires a tremendous amount of computational power, in addition to a deep understanding of forces driving parallel computers (Bianchini R. and Carrera E.V., 2001). Essentially, parallel computer architecture has matured to the point where it needs to be studied from a basis of engineering principles and quantitative evaluation of performance and cost.

Large-scale distributed memory and shared memory multiprocessor architectures are the most feasible way of achieving the enormous computational power required in many science and engineering applications (Chaudhuri et al., 2003). Such systems could be resized, skilled, integrated and developed to build very effectively super computers. Figure 1.1 depicts the architecture models for both shared memory and distributed memory.

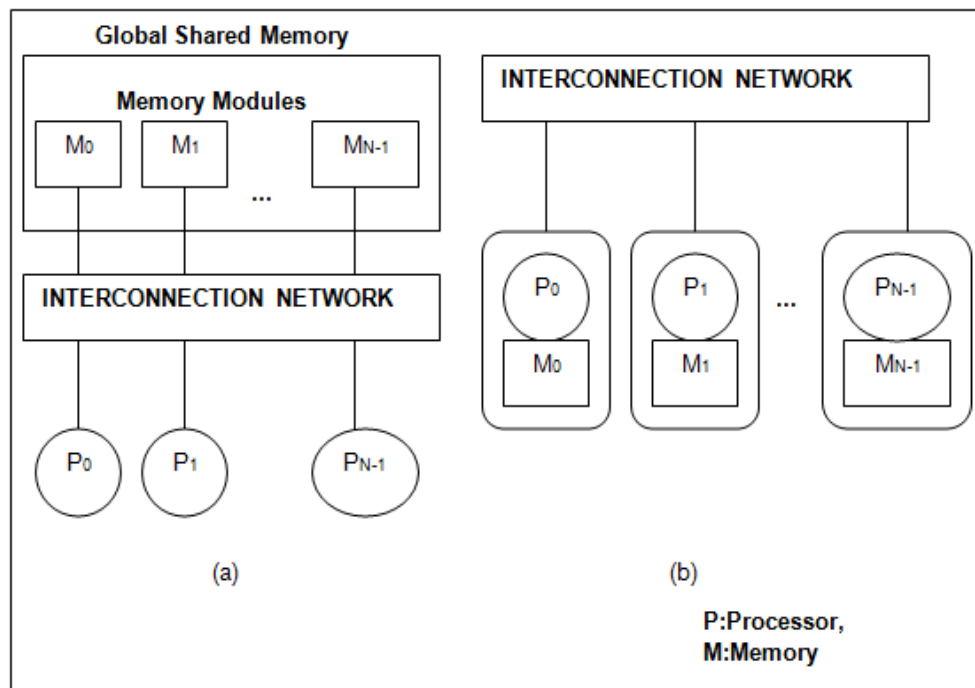


Figure 1.1. Shared-Memory v.s Distributed-Memory

Shared memory architecture combines programming advantages of shared-memory with scalability advantages of MP. In this paradigm the processors access all memory as global address space. It is burdened by the lack of scalability between memory and the CPUs and has a long average latency. On the other hand, in the distributed memory structure each processor has a private local memory and the memory is scalable with the number of processors. The access method is commonly known as the NUMA, which affects the elapsed times.

Parallel programming models are evolving apace and can truly utilize large-scale parallel computing systems. Several parallel programming models exist in common use and MP and shared memory programming models are the most popular ones.

In the MP model, a set of nodes use their own local memory during computation. Nodes exchange data through communications by sending and receiving messages and data transfer usually requires intensive cooperative operations to be performed by each process. A MP programming model uses a set of primitives that allows processes to communicate with each other. These include the *send*, *receive*, *broadcast* and *barrier* primitives. The send primitive takes a memory

buffer and sends it to a destination node. The receive primitive accepts a message from a source node and stores it in a specified memory buffer. The basic programming model used in MP architectures is based on the idea of matching a send request on one processor with a receive request on another. In such scheme, send and receive are blocking; that is, send blocks until the corresponding receive is executed before data can be transferred.

MP communication protocol supports end-to-end packet acknowledgment. For every packet sent by a source node, there is a returned acknowledgment after the packet has reached the destination node. This allows source nodes to discover packet loss. Automatic retransmission of a packet is made if the acknowledgment is not received within a preset time interval. A MP programming style is the preferred style for performance on such model. Also MP without acknowledgement protocol can be defined as above neglecting the fact that the source is not in need to learn whatever the sent packet has arrived or not. The main drawback of MP is the programmer's responsibility for determining and orchestrating all parallelism.

In the shared-memory programming model, tasks share a common address space, which they read and write asynchronously. An advantage of this model is that the notion of data "ownership" is lacking, so there is no need to specify explicitly the communication of data between tasks. Program development can often be simplified.

1.2. Motivation and the aim of the thesis

The performance analysis of a multiprocessor architecture is a very crucial factor in designing MP and DSM multiprocessor systems. Very often, simulation is the only feasible method because of the nature of the problem and because analytical techniques become too difficult to handle. Simulation occurs at many levels, from circuit to system and at different degrees of detail as the design evolves. Execution-driven and trace-driven multiprocessor simulations have been extensively used in order to obtain a reliable and accurate prediction of the final design. One of the problems with simulation is that although these simulations can be done at a high level of abstraction, they still are extremely time consuming. There are several

reasons why this is the case. First, the benchmarks that need to be simulated typically consist of several hundreds of billions of dynamically executed instructions. Second, multiple of these benchmarks need to be simulated in order to cover a representative set of applications. Third, the complexity of the target system reflects itself in the complexity of the simulator making the simulator at least four orders of magnitude slower than native hardware execution. Fourth, during design space exploration all benchmarks need to be simulated multiple times in order to identify the optimal design for a given cost function covering performance, power, area, cost, reliability, etc (Culler et al., 1999 and Kurose et al., 2010).

With the objective of reducing simulation time without losing accuracy, some interesting proposals have appeared in the last years. The first one is the sampled simulation, which chooses in an intelligent way a small portion of the program trace to simulate (Wenisch et al., 2006). The second one is using a reduced set of the inputs of a benchmark (Eeckhout et al., 2005). Finally, there is statistical modeling and simulation, which characterizes the behavior of the program and architecture with some probability distributions (Nussbaum and Smith, 2002; Genbrugge and Eeckhout, 2007).

A statistical simulation is a robust, flexible and suitable tool in multiprocessor design, but it can still be time consuming especially when DSM and MP multiprocessor systems to be simulated have many parameters and these parameters have to be tested with different probability distributions or values. Due to this problem, we propose to apply intelligent techniques for predicting the performance of a multiprocessor in a faster way. The basic idea is to collect several numbers of multiprocessor performance measures by using a statistical simulation and predict the performance of the MP and DSM system for a large set of input parameters based on these by using machine learning methods.

In this thesis, MFANN, GRNN, SVR and MLR techniques have been used to predict the performance measures of the SOME-Bus architecture employing both the MP and DSM programming models. The protocols used are: MP with ACK's , MP without ACK's and DSM protocols. OPNET Modeler (Opnet Inc., 2012) is used to statistically simulate the SOME-Bus architecture. The input variables of the

prediction model include T/R , node number, thread number, traffic pattern and protocol type. The output variables of the prediction model include averages for: CWT , CU , NRT , PU and IWT . The performance of the prediction models have been evaluated by calculating their SEE , R , MAE , RAE and $RRSE$ error values. In summary, it is shown that MFANN's perform better than GRNN, SVR and MLR for predicting the performance measures of a multiprocessor architecture.

1.3. Organization of the thesis

The rest of this thesis is organized as follow: Chapter 2 presents an overview of the SOME-Bus architecture. Chapter 3 gives an overview of the methods applied. Chapter 4 describes simulation framework and dataset generation. Results and a detailed discussion of the findings are presented in Chapter 5. Finally, Chapter 6 concludes the thesis.

1.4. Literature Review

Advances in optical technology combined with daemon intelligence in neural networks have promoted the parallel multiprocessor interconnection network as a realistic, competitive and a highly recommended candidate to face the high quest for super power systems (Wolf Marilyn, 2012).

Simulation is an indispensable way for building a multiprocessor system (Yi et al., 2006). It enables one to quickly analyze the behavior of a complex system and to evaluate subtle design trade-offs in a controlled experimental environment. Trace-driven simulation is a commonly used a simulation techniques when traces are prepared and fast simulation is required especially in an early design stage. Trace-driven simulation's increased speed results from replacing the detailed functional execution of a benchmark with a highly representative trace of a program execution. The trace may capture every executed instruction of a program, or it may contain the information of certain events, such as L2 cache accesses (Uhlig et al., 1997). Trace

driven simulation with a full instruction trace is a widely used method to precisely model the performance of an out-of-order superscalar processor (Black et al., 1996).

(Black et al., 1996) showed that sampling techniques present a problem to the accuracy of trace-driven simulation for multiprocessors system. Whereas, (Lee K. and Cho S., 2012) advocated using timing-embedded filtered trace accurately models superscalar processor performance. Much trace-driven simulation work has focused on either tracing memory references (Uhlig et al., 1997) or using a full trace of executed instructions for relatively fast simulation with complete fidelity.

In paper (Cao et al., 2000) a simulation system for load balancing algorithms is constructed on a local area network of DEC workstations, which directly executes the codes of the load balancing algorithms but simulates the underlying network and system environment. Using the simulation system, simulations with real workload distribution are conducted. Traces of user workstation activity collected in a university department environment are used in the simulation runs. In that paper authors described the methods used for distributed direct execution simulation of load balancing algorithms and the simulation results are discussed.

Investigation in (Chung et al., 2001) analyzed the collective performance of different settings of the CC-NUMA multiprocessor architecture. In that study the simulation was used and the results showed that the bottle-neck on the system resources subsystem could be identified and effectively removed by setting the configurations. Also, (Chou et al., 2004) proposed a simulation technique based on the epoch model to quickly derive the memory-level parallelism of a program. Their simulator is a very simplified processor model based on several assumptions. Nonetheless, the simulator shows accurate results, especially when a long off chip access latency is assumed. In (Fang et al., 2005) an execution-driven simulation used to quantitatively compare the performance of a variety of synchronization mechanisms based on both existing hardware techniques and active memory operations was considered.

In (Rui et al., 2007) a dynamic pre-fetching thread scheme is proposed to accelerate sequential programs on chip multiprocessors. The evaluation was performed by using a detailed cycle accurate execution-driven simulator. In order to

demonstrate the performance potential of the architecture, dual core configuration was used in the simulation. The train sets were used for SPEC benchmarks to achieve reasonable simulation times. The study argues that for a set of memory limited benchmarks selected from Olden benchmark, SPEC CPU2000 as well as stream benchmark, an average speed up of 3.8% is achieved on dual-core CMP when constructing basic dynamic pre-fetching threads and this gain grows to 29.6% when adopting its aggressive thread construction policies.

In summary, there are many trace-driven multiprocessor simulators, (Lee et al., 2010) introduced a two-phase trace driven simulation using fast multiprocessor architecture simulation based software. In (Li et al., 2006), that also use timing-embedded filtered traces.

With the advent of multiprocessor systems and their ever-increasing complexity, the software simulation strategy based on instruction set simulators is no longer efficient enough for exploring the large design space of multiprocessor systems in early design phases. Motivated by the limitations of instruction set simulators, a lot of recent research activities focused on software simulation strategies based on native execution (Wang et al., 2010). The main contribution of the study was introduced a new software performance simulation approach, called iSciSim which achieves high estimation accuracy, high simulation speed and low implementation complexity.

In (Bani-Mohammad et al., 2011), authors evaluate Adaptive Noncontiguous Allocation for different communication patterns using an event-driven simulator operating at the flit level. This allows for a more realistic evaluation that takes into account the shape of allocation and contention among messages. Also, the authors have carried out extensive simulation experiments so as to compare the performance of several promising noncontiguous allocation strategies proposed for 2D mesh-connected multicomputer.

In (Lee K. and Cho S., 2012) trace-driven simulation of superscalar processors is carried out. The authors describe and comprehensively evaluate the pairwise dependent cache miss model (PDCM), a framework for fast and accurate trace-driven simulation of out-of-order superscalar processors. The model determines

how to treat a cache miss with respect to other cache misses recorded in the trace by dynamically reconstructing the reorder buffer state during simulation and honoring the dependencies between the trace items. The authors arguing that a PDCM-based simulator produces highly accurate simulation results (less than 3% error) with fast simulation speeds ($62.5\times$ on average) compared with an execution-driven simulator. Also, the authors claimed that the proposed simulation method is capable of preserving a processor's dynamic off-core memory access behavior and accurately predicting the relative performance change when a processor's low-level memory hierarchy parameters are changed.

Many proposals evaluating the performance of a multiprocessor architecture have been extensively studied in literature in the domain of high-performance parallel computing (Katsinis, 1998; Cohen et al., 2000; Katsinis, 2001; Hecht, 2002; Nussbaum and Smith, 2002; Zhu et al., 2004; Eeckhout et al., 2005; Wenisch et al., 2006; Akay and Katsinis, 2007; Genbrugge and Eeckhout, 2007). However, only five papers showed that employing machine learning techniques can be used to predict the performance measures of a large-scale multiprocessor interconnection network.

In (Akay and Abasikeleş, 2010), a broadcast-based multiprocessor architecture called the SOME-Bus employing the DSM programming model was considered. The statistical simulation of the architecture was carried out to generate the dataset. The dataset contained the following variables: ratio of the mean message channel transfer time to the mean thread run time (T/R), probability that a block can be found in modified state (P_m), probability that a data message is due to a write miss (P_w), probability that a cache is full (P_{cf}) and probability of having an upgrade ownership request (P_{uor}). Support vector regression was used to build prediction models for predicting average network response time (NRT), average channel waiting time (CWT) and average processor utilization (PU). It was concluded that support vector regression (SVR) model is a promising tool for predicting the performance measures of a distributed shared-memory multiprocessor.

The following papers have been published by using some of the material that also appear in this thesis.

In (Akay and Zayid, 2011) and (Zayid and Akay, 2012a) MFANN models were developed to predict the measures of the SOME-Bus architecture employing the MP with ACK's and the hybrid MP protocols, respectively. In the first study, only the MFANN models were developed and the performance of the models was evaluated by calculating the error metrics *MAE*, *RMSE*, *RAE* and *RRSE*. In the second paper, only the values for *SEE* and *R* are calculated and the results of the MFANN-based models were compared with the ones obtained by GRNN, SVR and MLR model. Both papers concluded that MFANN models shortens the time quite a bit for obtaining the performance measures of a MP multiprocessor and can be used as an effective tool for this purpose.

In (Zayid and Akay, 2012b), authors developed a MFANN model to predict the performance measures of the SOME-Bus architecture employing the MP programming model with ACK's. OPNET Modeler (Opnet Inc., 2012) was used to statistically simulate the MP on the SOME-Bus architecture. The input variables of the prediction model include *T/R*, node number, thread number and traffic pattern, where as the output variables of the prediction model include averages for *CWT*, *CU*, *NRT*, *PU* and *IWT*. The performance of the prediction models have been evaluated by calculating their *SEE* and *R* values. The study compared the results of the MFANN-based model with the ones obtained by GRNN-based, SVR-based and MLR-based models. It was shown that MFANN's perform prediction better than GRNN-based, SVR-based and MLR-based models.

In (Zayid and Akay, 2012c) authors developed MFANN models for predicting the performance measures of a multiprocessor architecture interconnected by the SOME-Bus, which employs the MP with no ACK's. OPNET Modeler was used to simulate behavior of the SOME-Bus multiprocessor architecture and to create the datasets. Several machine learning techniques have been used. The results show that MFANN-based model gives the best results (i.e. lowest *SEE* and highest *R*) among all predicting models. It was concluded that MFANN models shorten the time quite a bit for obtaining the performance measures of a MP multiprocessor.

2. OVERVIEW OF THE SOME-BUS ARCHITECTURE

2.1. The SOME-Bus Architecture

Demanding for the zero latency and high bandwidth multiprocessor interconnection network topology that provides super power is very desirable for parallel computing applications (Kulick et al., 1995 and Aci et al., 2010).

SOME-Bus (Simultaneous Optical Multiprocessor Exchange Bus) is a processor interconnection scheme that uses the properties of optics to provide the benefits of small interconnection distances and high data rates (Katsinis, 2001). It is a proposed optical interconnection architecture for over a hundred processors which contains a dedicated transmission channel for each processor to eliminate global arbitration and to provide bandwidth that scales with the number of processors in the machine. Unlike electrical buses in which the limits are due to the electrical characteristics of the wire, the bandwidth of optical interconnects is not limited by the fiber optics used to connect the transmitters and receivers; the bandwidth limitations are due to the transmitter and receivers (Cohen, Hyde and Gaede, 2000).

SOME-Bus is low-latency, high-bandwidth, fiber-optic network which directly connects each processing node to all other nodes without contention. One of its key features is that each of P nodes has a dedicated broadcast channel which can operate at several Gbytes/second, depending on the configuration. In general, the SOME-Bus contains K fibers, each carrying M wavelengths organized in M/W channels, where each channel is composed of W wavelengths. The total number of fibers is $K = PW/M$. A simple configuration with 128 nodes ($P = 128$ channels) and $W = 1$ wavelength per channel would require $K = 32$ fibers with $M = 4$ wavelengths per fiber and a receiver array at each node containing 128 detectors organized as 32×4 over the surface of a single chip. Each of P nodes also has an input channel interface based on an array of P receivers (each with W detectors) which simultaneously monitors all P channels (Katsinis, 2004).

The physical implementation of SOME-Bus is motivated by recent progress in optical communication, dense-wavelength-division-multiplexing and

optoelectronics. Slant Bragg gratings (Bouzid and Abushagur, 1996) are written directly into the fiber core and are used as narrow-band, inexpensive output couplers. This coupling of the evanescent field allows the traffic to continue and eliminates the need for regeneration. Figure 2.1 shows the parallel receiver array and output coupler. The SOME-Bus also uses amorphous silicon (a-Si) photo-detectors built as super structures on the surface of electronic processing devices.

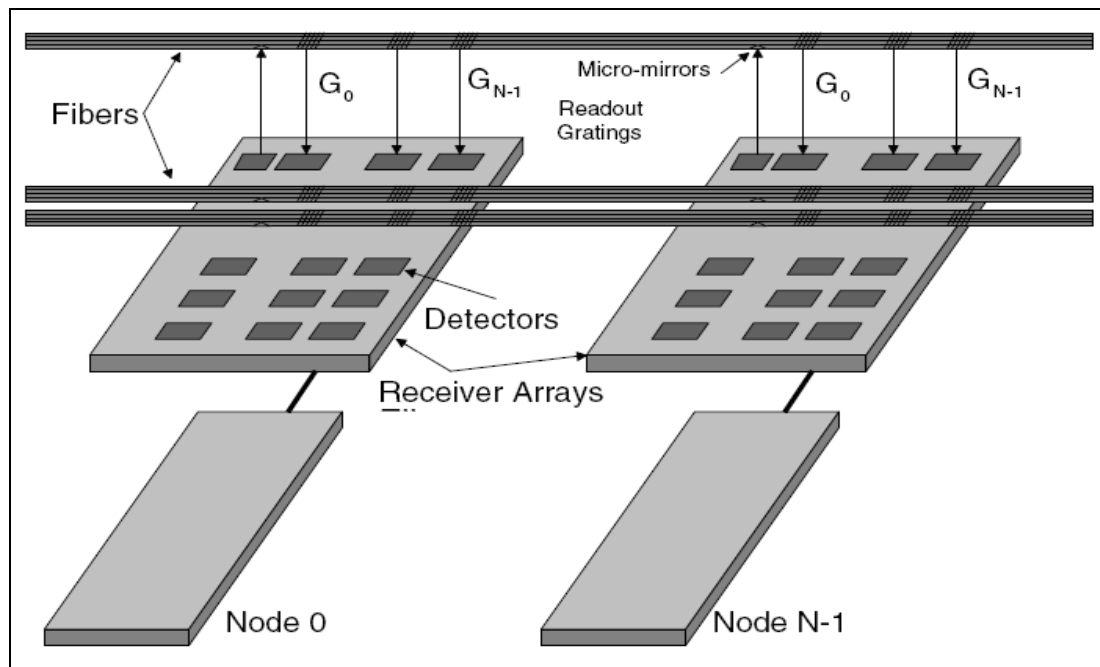


Figure 2.1. Parallel receiver array (Katsinis, 2004)

The receiver array does not perform any routing and consequently its hardware complexity is small. It contains an optical interface which performs address filtering, barrier processing, length monitoring and type decoding. If a valid address is detected in the message header, the message is placed in a queue, otherwise the message is ignored. The address filter can recognize multicast group addresses as well as broadcast addresses in addition to recognizing the address of the host node. The receiver array also contains a set of queues such that one queue is associated with each input channel, allowing messages from any number of nodes to arrive and be buffered simultaneously. This organization supports multiple simultaneous broadcasts, provides bandwidth that scales directly with the number of nodes in the

system and eliminates the need for global arbitration. Arbitration may be required only locally in the receiver array when multiple input queues contain messages (Hecht and Katsinis, 2003).

Once the logic level signal is restored from the optical data, it is directed to the input channel interface which consists of two parts: the optical interface and the processor interface. Figure 2.2 shows the optical interface which includes physical signaling, address filtering, barrier processing, length monitoring and type decoding (Zhu et. al., 2004). Each receiver generates a data stream which is examined to detect the start of the packet and the packet header. The header decode circuitry examines the header field, which includes information on the message type, destination address (or addresses) and length, to determine whether or not the message is a synchronization message. If the message is a synchronization message, it is handled by the barrier circuitry, otherwise the destination address is compared to the set of valid addresses contained in the address decode circuitry. In addition to recognizing the local node address, the address filter can recognize multicast group addresses as well as broadcast addresses. Once a valid address has been identified, the message is placed in a queue. If the address does not match, the message is ignored (Katsinis, 2004).

Figure 2.3 shows the processor interface which includes a routing network (resolver circuit) and a queuing system. One queue is associated with each input channel, allowing messages from any number of processors to arrive and be buffered simultaneously, until the local processor is ready to remove them. The resolver circuit receives a request signal (R_{in}) from each non-empty queue and produces the index of the next queue to be accessed under either the limited or the exhaustive service disciplines.

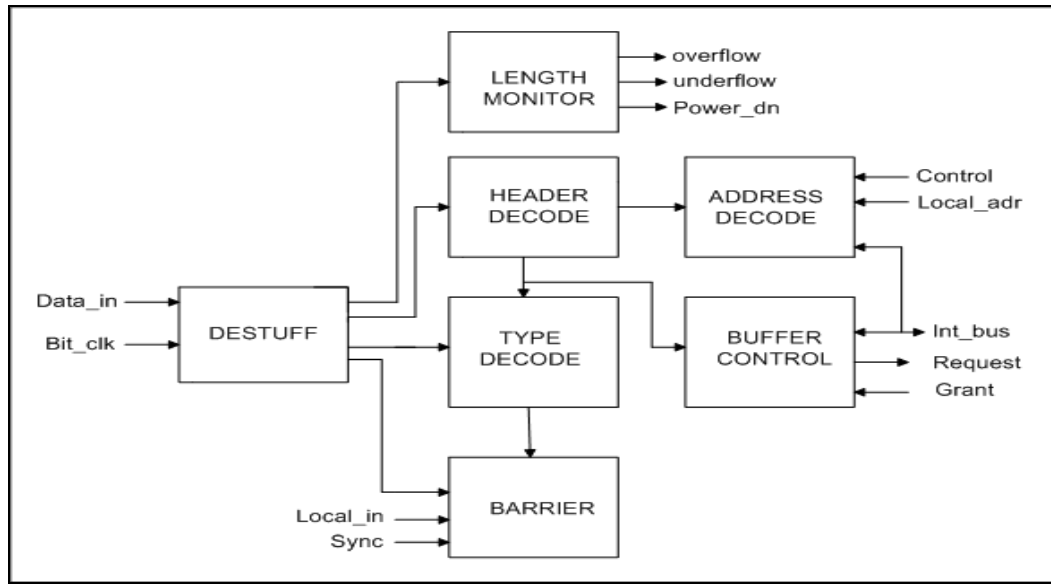


Figure 2.2. The SOME-Bus Optical Interface (Zhu et al., 2004)

The local processor can force the next queue selection through the Pin input. A straightforward implementation of the resolver as a selection tree, using logic gates to select the next queue and multiplexers to forward the corresponding queue index, requires only several hundred gates organized in $\log_2(P)$ levels. The time required to select the next queue (polling walk time) is consequently very small and can be overlapped with the queue access time (Katsinis, 2004).

The SOME-Bus has much more functionality than plain crossbar architecture. With N nodes, the diameter of the SOME-Bus is 1, the time needed for all-to-all communication with distinct messages is $O(N)$ and the time needed for synchronization is $O(1)$. Unlike a fully-connected point-to-point network, where the number of transmitters and channels increases $O(N^2)$, the number of transmitters and channels of the SOME-Bus is $O(N)$, quite smaller than the number required in other popular architectures, such as the hypercube or the torus.

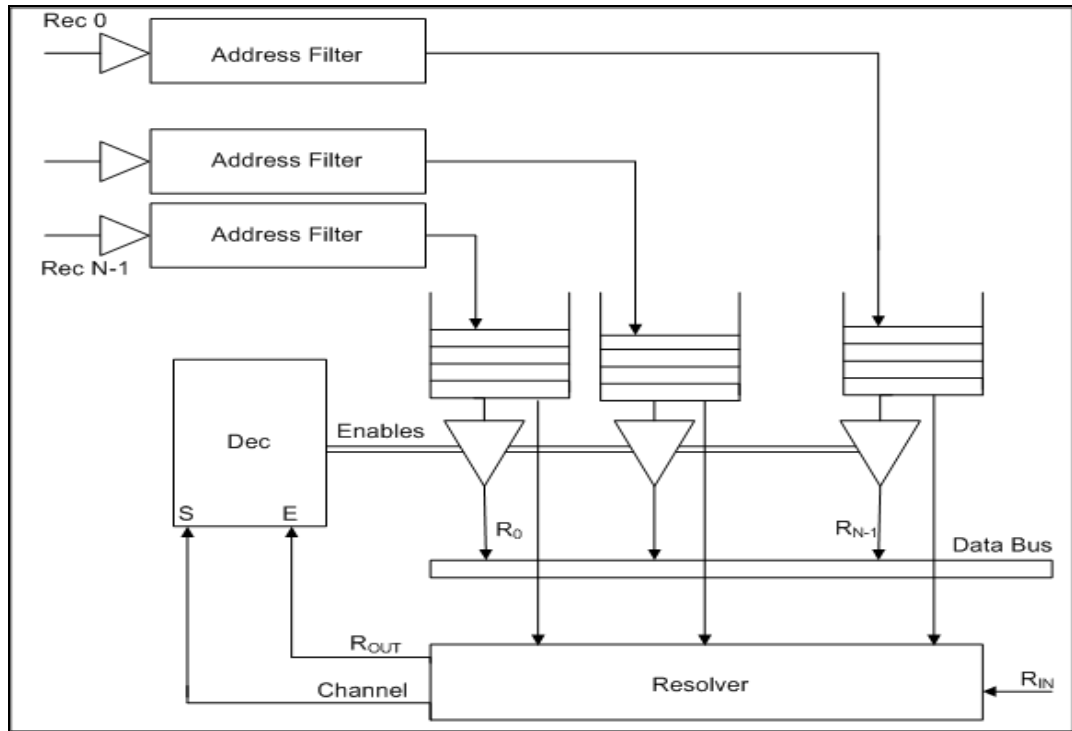


Figure 2.3. The SOME-Bus processor interface (Katsinis, 2004)

The total number of receivers is N^2 , which is larger than the number required in other architectures. They are arranged so that N receivers are fabricated as A-Si structures constructed as a thin film directly on the surface of a digital CMOS device, with no lithography required. Because of the low conductivity of the amorphous silicon layer, no subsequent patterning is required and therefore the yield and cost of the receiver is determined by the yield and cost of the CMOS device itself. The full receiver array can be implemented on a single chip even for large values of N ($N > 128$). Therefore, the total receiver cost is approximately $O(N)$ instead of $O(N^2)$. The SOME-Bus with N nodes can be scaled to $2N$ nodes by using four SOME-Bus segments to create twice the number of channels where each channel is twice as long to accommodate the additional nodes (Hecht and Katsinis, 2003).

3. OVERVIEW OF METHODS

3.1. Multi-layer Feed-Forward Artificial Neural Networks

The MFANN employs the model structure of a neural network which is a powerful computational technique for modeling complex non-linear relationships particularly in situations where the explicit form of the relation between the variables involved is unknown (Alpaydm E, 2010; Chen M-S and Yen H-W, 2011). A MFANN consists of at least three layers, input, output and hidden layer. The schematic diagram of a MFANN is shown in Figure 3.1 each neuron in a layer receives weighted inputs from a previous layer and transmits its output to neurons in the next layer. The summation of weighted input signals are calculated by Eq. (3.1.) and this summation is transferred by a nonlinear activation function given in Eq. (3.2.). The results of the network are compared with the actual observation results and the network error is calculated with Eq. (3.3.). The training process continues until this error reaches an acceptable value (Khashei et al., 2012).

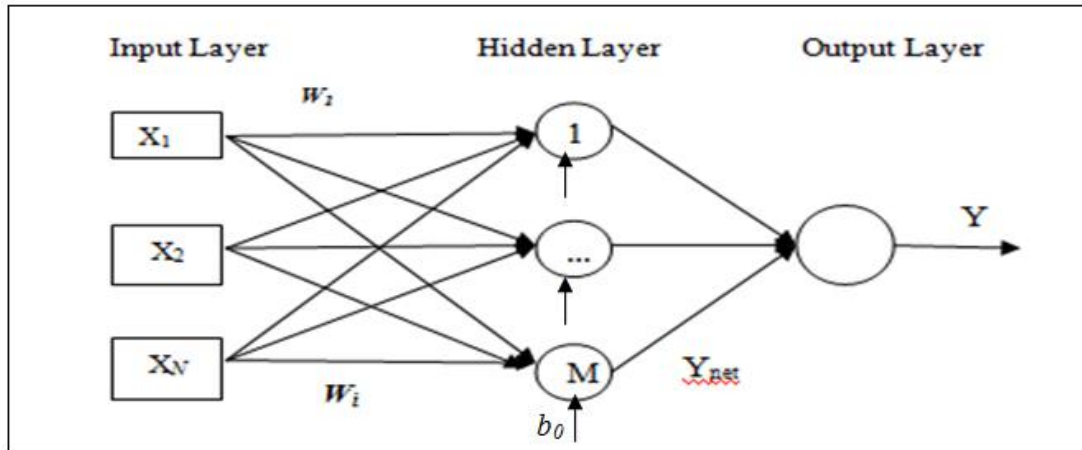


Figure 3.1. A typical multilayer feed-forward Neural Network

$$Y_{net} = \sum_{i=1}^n X_i w_i + b_0 \quad (3.1.)$$

$$Y = \sum_{i=1}^M f(Y_{net}) = \sum_{i=1}^M \left(\frac{1}{1 + e^{-Y_{neti}}} \right) \quad (3.2.)$$

$$J_r = \frac{1}{2} \sum_{i=1}^k (Y_i - O_i)^2 \quad (3.3.)$$

Y_i is the response of neuron i , $f(Y_{net})$ is the nonlinear activation function, Y_{net} is the summation of weighted inputs, X_i is the neuron input, w_i is the weight coefficient of each neuron input, b_0 is the bias, J_r is the error between observed value and network response, O_i is the observed value of neuron i . Also, N is the number of input variables and M is the number of the hidden neurons in hidden layer.

3.2. Generalized Regression Neural Networks

The GRNN is a generalization of both radial basis function networks and probabilistic neural networks that can perform linear and nonlinear regression (Specht, 1991; Firat and Gungor, 2009). These feedforward networks use basis function architectures that can approximate any arbitrary function between input and output vectors directly from training samples and they can be used for multidimensional interpolation (Wachowiak, 2001). The main function of a GRNN is to estimate a linear or nonlinear regression surface on independent variables (input vectors) X , given the dependent variables (desired output vectors) Y . That is, the network computes the most probable value of an output, O_x , given only training vectors X . Specifically, the network computes the joint probability density function of X and Y . The expected value of Y given X is expressed as (Specht, 1991; Wachowiak, 2001; Firat and Gungor, 2009):

$$E[Y / X] = \frac{\int_{-\infty}^{\infty} Y f(X, Y) dy}{\int_{-\infty}^{\infty} f(X, Y) dy} \quad (3.4.)$$

An important advantage of the GRNN is its simplicity and fast approximation procedure. Another attractive feature is that, unlike back propagation-based neural networks, GRNN does not converge to local minima (Specht, 1991). The topology of a GRNN consists of four layers. Figure 3.2 shows the GRNN layers architecture.

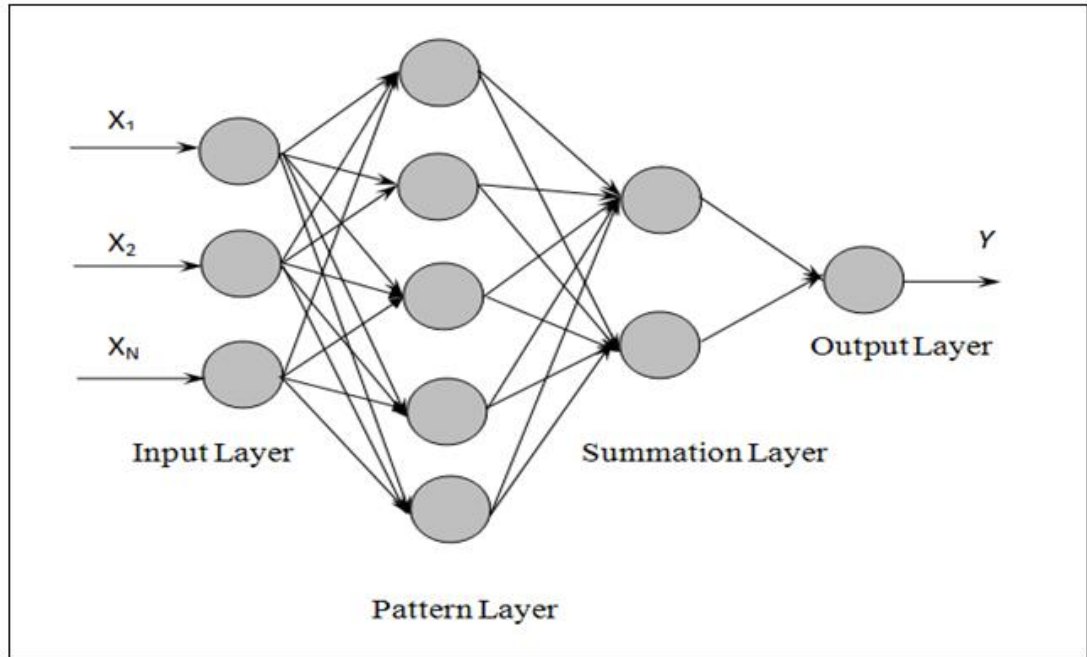


Figure 3.2. Architecture of GRNN model.

First, there is an input layer that is fully connected to the pattern layer. Second, there is a pattern layer that has one unit for each pattern. It computes the pattern Gaussian function expressed by

$$h_i = \exp[-D_i^2 / 2s^2]; \quad (3.5.)$$

where

$$D_i^2 = (x - X_i)^T (x - X_i) \quad (3.6.)$$

s denotes the smoothing parameter, x is the input presented to the network and X_i is each of the training vector. Third, there is a summation layer that has two units N and P . The first unit computes the weighted sum of the hidden layer outputs. The second unit has weights equal to “1” and therefore sums exponential terms (h_i) alone. Fourth, there is an output unit that divides N by P to provide the desired prediction result.

3.3. Support Vector Regression

3.3.1. Linear Support Vector Regression

Assume given the training data $(x_i, y_i), (i = 1, \dots, l)$, where x is a d -dimensional input with $x \in \mathcal{R}^d$ and the output is $y_i \in \mathcal{R}$. The linear regression model can be written as follows (Vapnik, 2000):

$$f(x) = \langle w, x \rangle + b, \quad w, x \in \mathcal{R}^d, \quad b \in \mathcal{R}, \quad (3.7.)$$

where $f(x)$ is an unknown target function and $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{R}^d$.

In order to measure the empirical risk (Cherkassky et al., 2004) the study should specify a loss function. The most common loss function is the ε -insensitive loss function proposed by Vapnik (Vapnik, 2000) and is defined by the following function:

$$L_e(y) = \begin{cases} 0 & ; \text{for } |f(x) - y| \leq \varepsilon \\ |f(x) - y| - \varepsilon & ; \text{otherwise} \end{cases} \quad (3.8.)$$

The optimal parameters $\bar{w}$ and $\bar{b}$ in (3.7.) are found by solving the primal optimization problem (Gunn S R, 1998):

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (x_i^- + x_i^+) \quad (3.9.)$$

with constraints:

$$\begin{aligned} y_i - \langle w, x_i \rangle - b &\leq e + x_i^+, \\ \langle w, x_i \rangle + b - y_i &\leq e + x_i^+, \\ x_i^+, x_i^- &\geq 0, \quad i=1, \dots, l \end{aligned} \quad (3.10.)$$

where C is a pre-specified value that determines the trade-off between the flatness of $f(x)$ and the amount up to which deviations larger than the precision e are tolerated. The slack variables x_i^- and x_i^+ represent the deviations from the constraints of the e -tube.

Usually the dual problem is solved. The corresponding dual optimization problem is defined as

$$\max_{a, a^*} -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l (a_i^* - a_i)(a_j^* - a_j) \langle x_i, x_j \rangle - \sum_{i=1}^l y_i (a_i^* - a_i) - e \sum_{i=1}^l (a_i^* + a_i) \quad (3.11.)$$

with constraints:

$$\begin{aligned} 0 &\leq a_i, a_i^* \leq C, \quad i=1, \dots, l \\ \sum_{i=1}^l (a_i - a_i^*) &= 0 \end{aligned} \quad (3.12.)$$

Solving the optimization problem defined by (3.11.) and (3.12.) gives the optimal Lagrange multipliers a and a^* , while $\bar{w}$ and $\bar{b}$ are given by

$$\begin{aligned}\bar{w} &= \sum_{i=1}^l (a_i^* - a_i) x_i, \\ \bar{b} &= -\frac{1}{2} \left\langle \bar{w}, (x_r + x_s) \right\rangle,\end{aligned}\tag{3.13.}$$

where x_r and x_s are support vectors (Gunn S R, 1998).

3.3.2. Non-linear Support Vector Regression

For nonlinear regression problems, a nonlinear mapping f of the input space onto a higher dimension feature space can be used and then linear regression can be performed in this space (Schölkopf and Smola, 2002). The nonlinear model is written as:

$$f(x) = \langle w, f(x) \rangle + b, \quad w, x \in \mathfrak{R}^d, \quad b \in \mathfrak{R}, \tag{3.14.}$$

where

$$\begin{aligned}\bar{w} &= \sum_{i=1}^l (a_i - a_i^*) f(x_i), \\ \left\langle w, \bar{f}(x) \right\rangle &= \sum_{i=1}^l (a_i - a_i^*) \langle f(x_i), f(x) \rangle = \sum_{i=1}^l (a_i - a_i^*) K(x_i, x), \\ \bar{b} &= -\frac{1}{2} \sum_{i=1}^l (a_i - a_i^*) (K(x_i, x_r) + K(x_i, x_s))\end{aligned}\tag{3.15.}$$

Where x_r and x_s are support vectors. Note that we express dot products through a kernel function K that satisfies Mercer's conditions (Vapnik, 2000). Equation (3.15.) can be written as follows if the term $\bar{b}$ is accommodated within the kernel function:

$$\sum_{i=1}^l (a_i - a_i^*) K(x_i, x) \tag{3.16.}$$

Several kernel functions have appeared in literature. The radial basis function (RBF) has received significant attention, most commonly with a Gaussian of the form:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2r^2}\right). \quad (3.17.)$$

where r is the width of the RBF kernel.

3.4. Multiple Linear Regression

The multiple linear regression models are extension of a simple linear regression model to incorporate two or more explanatory variable in a prediction equation for a response variable. Multiple regression modeling is now a mainstay of statistical analysis in most fields because of its power and flexibility. It requires very little effort (and sometimes even less thought) to estimate very complicated models with large numbers of variables. In multiple regression the general model is as:

$$y_i = B_0 + B_1x_{i1} + B_2x_{i2} + \dots + B_nx_{in} + E_i \quad (3.18.)$$

where $i = 1, 2, \dots, n$; B_i is the residual, E_i is the difference between the value of the dependent variable predicted by the model and the dependent variable, x is the independent parameter.

MLR takes a group of random variables and tries to find a mathematical relationship between them. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points. The study can rewrite the first section on the right-hand side of equation (3.18.) as

$$LP_i = B_0 + B_1x_{i1} + B_2x_{i2} + \dots + B_nx_{in} \quad (3.19.)$$

where LP_i is known as the *linear predictor* and it is the value of y_i predicted by the input variables. The difference $y_i - LP_i = E_i$ is the error term.

The models are fitted by choosing estimates $B_0 + B_1 + B_2 + \dots + B_n$, which minimize the sum of squares of the predicted error. These estimates are termed ordinary least squares estimates. Using these estimates the study can calculate the fitted values y_i^{fit} and the observed residuals $e_i = y_i - y_i^{fit}$. Here it is clear that the residuals estimate the error term (Draper and Smith, 1998). MLR has wide areas of usage and can be summarized as follow:

1. To adjust the effects of an input variable on a continuous output variable for the effects of confounders. This is commonly known as analysis of covariance.
2. To analyze the simultaneous effects of a number of categorical variables on an output variable.
3. To predict a value of an outcome, for given inputs. In this study we applied MLR to perfectly predict performance measures of a multiprocessor network.

4. SIMULATION AND DATASET GENERATION

4.1. Simulation Framework

OPNET Modeler (OPNET Technologies Inc., 2012) is an environment for network modeling and simulation, which can also be used for designing and studying interconnection networks and protocols. It is based on a series of hierarchical editors, project editor, node editor and process editor, which directly parallel the structure of interconnection networks and protocols. The node editor captures the architecture of a system by depicting the flow of messages between functional elements, called modules. Each module can generate, send and receive packets from other modules to perform its function within the node. Modules typically represent physical resources such as buffers, ports, queues and buses. Modules are assigned process models, developed in the process editor, to achieve any required behavior. The process editor uses a finite state machine approach to support specification of protocols, algorithms and queuing policies. States (the condition of a module) and transitions (a change of state) graphically define the progression of a process in response to events. States have “enter executives” (code that is executed when the module moves into a state) and “exit executives” (code that is executed when the module leaves a state), and there is “transition executive” (code that is executed in response to a specific event). There are two kinds of states: An unforced (red) state is the one that returns control of the simulation to the simulation kernel after executing its enter executives. A forced (green) state is one that does not return control, but instead immediately executes the exit executives and transitions to another state.

4.2. MP Framework and Dataset Generation

OPNET Modeler (Opnet Inc., 2012) is used to simulate the SOME-Bus architecture employing the MP protocol with and without ACK's. Figure 4.1 shows the node model of the simulated architecture. Each node contains a processor

station in which the incoming messages are stored and processed and also a channel station in which the outgoing messages are stored before transferring them onto the network.

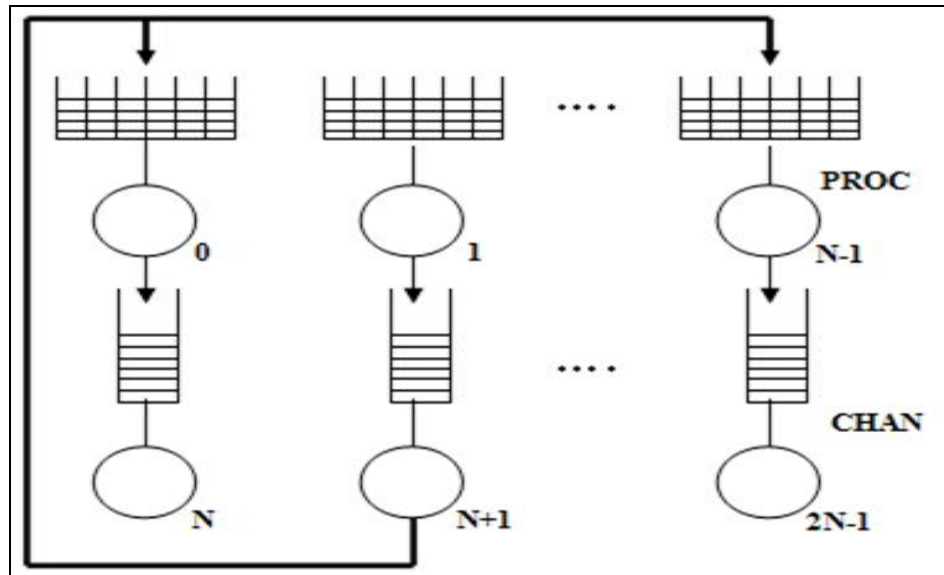


Figure 4.1. A typical N-node SOME-Bus architecture using MP protocols.

The underlying process model that controls queue modules' behavior is OPNET's built-in `acb_fifo` model which is shown in Figure 4.2. The model has its own server and can concentrate multiple incoming packets streams into its single internal queuing resource. It also supports the First-in-First-out service ordering discipline and a way to control service times. The “init” state is used to initialize the process and setting the appropriate variables. If a packet arrives when the process is in “init” state, the process transitions to the “arrival” state, else it transitions to the “idle” state where it waits for packet arrival. The “arrival” state is used for receiving packets and starting service. In the “arrival” state, if the server is not busy then the process moves into the “svc_start” state, which in turn transitions to the “idle” state, where it waits either for packet arrival or service completion. While in the “idle” state, if the processing of a packet is completed, the process moves into the “svc_compl” state. While in the “svc_compl” state, if the queue is not empty, the process moves into the “svc_start” state.

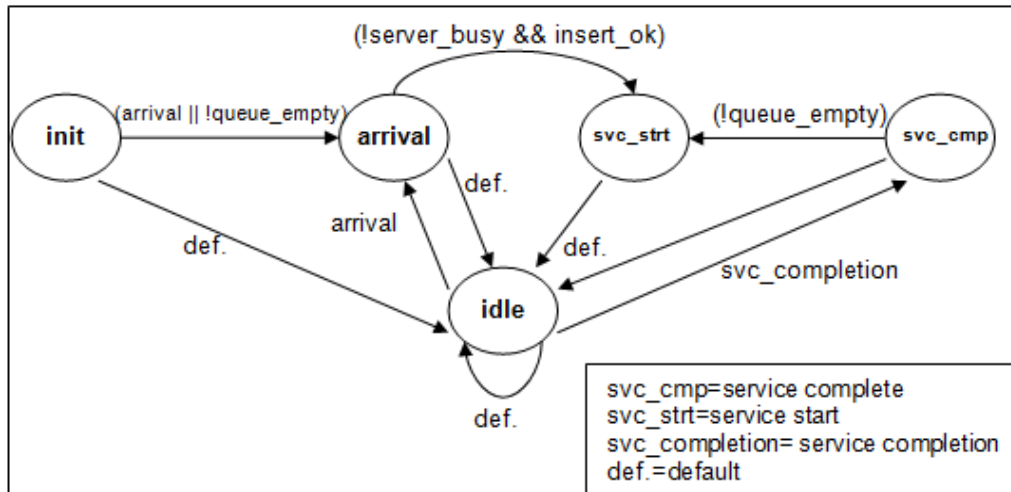


Figure 4.2. A typical process model for the queues.

Using synthetic traffic workloads and running a simulator for a large number of cycles to get performance results with the network in steady state has been widely used in past studies (Alonsoa, Izu and Gregorioc, 2008). Although not a completely realistic assumption, the results obtained with synthetic traffic are expected to indicate the minimum level of performance the network could provide under actual traffic. This has been shown to be true for some applications such as Radix or LU (Singh, Weber and Gupta, 1992) which are part of the SPLASH benchmark suite. A synthetic traffic workload is defined by three important parameters: *spatial distribution* describes the destination node distribution for each source node, *temporal distribution* specifies packet generation times and *message length distribution* gives the size of each message. Regarding spatial distributions, the study used a collection of well-known permutations: BR and PS. Thesis also included UN and HR traffic models.

Uniform traffic pattern can be represented by a traffic matrix, where each matrix element $\lambda_{s,d}$ gives the fraction of traffic sent from node s destined to node d . In the UN traffic, the destination node is selected using uniform distribution with mean in range from 1 to N . Bit permutations such as BR and PS are those in which each bit d_i of the b -bit destination address is a function of the one bit of the source address (Dally and Towels, 2004). In the HR pattern, the destinations of the 25% of the packets are chosen randomly within a small hot-region consisting of 12.5% of

the nodes (Blumrich et al., 2003) Table 4.1 lists the destination node selection for these traffic patterns.

Table 4.1. Synthetic traffic patterns

Name	Traffic Pattern
UN	$\lambda_{s,d} = 1/N$
BR	$d_i = b_{i+1}$
PS	$d_i = s_{i-1} \bmod b$
HR	The 25% of the packets are sent to 12.5% of the node group

Temporal distribution of packet generation can be implemented by independent or non-independent traffic sources (Alonsoa, Izu and Gregorioc, 2008). As its name implies, independent traffic sources progress independently of the others and may use a Poisson distribution or on-off models. Most simulation-based studies of interconnection networks use independent traffic sources (Shin and Pinkston, 2003). The main drawback of using just independent sources is that the obtained results may not be realistic representative of network performance under heavy loads (Izu, Alonso and Gregorio, 2005). Also, independent sources cannot capture reactive data exchange patterns, which are common in real applications. Non-independent traffic sources can simulate reactive data exchange patterns such as client-server traffic. In the simulations, the thesis utilized client-server traffic (i.e. a server node sends packets to respond to the reception of packets from clients) and used hybrid traffic sources (i.e., initially, all nodes generate traffic independently of the others, as time progresses traffic generation at the source / destination nodes depend on the receipt of messages from destination / source nodes). The processing time (R) is assumed to be exponentially distributed with a mean of 100 clock cycles.

The message transfer time (T) is assumed to be uniformly distributed with mean in range from 5 to 100 clock cycles. Since T is closely related to the packet length, using different values for T allows us to evaluate the performance of the

congestion control algorithm for varying packet sizes. The ratio T/R varies between 0.05 and 1. This range of the ratio is sufficient to capture the system behavior under most common configurations and cache behavior. Specifically, let m be the miss rate and F the number of instructions per second performed by the processor at each node. Also, let S be the mean packet size in bytes and C the channel bandwidth (in bytes per second). Then, the ratio of the mean thread run time to the mean packet transfer time $T/R = mSF/C$. In current high performance architectures, the ratio of F/C is in the range of 0.5 – 1. For example, in Cray XT3 (Hemenway, 2008) $F = 4.8 \times 10^9$, and the links have a peak bandwidth of 7.6 GB/s. With small cache blocks and miss rate in the neighborhood of 10% or less (due to the fact that programmers are going to target and distribute their applications for maximum locality, thus most accesses on well behaved applications are going to fall in cache), the resulting ratio T/R is in the range of 0.05 to 1.

The important parameters of the simulation are the number of *nodes* (selected as 16, 32 and 64), the number of the *threads* executed by each processor (ranging from 1 to 6), T/R , *thread run time* (exponentially distributed with a mean value of 100), and *traffic pattern* (i.e., UN, HR, BR, and PS).

The dataset obtained as a result of the simulation contains four input and five output variables. The input variables of the prediction model include T/R , *node number*, *thread number*, *traffic pattern* and protocol type (in case of hybrid MP). The output variables of the prediction model include average *CWT* (i.e. the time interval between the instant when a packet is enqueued in the output channel until the instant when the packet goes under service), average *CU* (i.e. average fraction of time that the channel server is busy), average *NRT* (i.e. the time interval between the instant when a message is enqueued in the output channel until the instant when the corresponding acknowledge message arrives at the input queue), average *PU* (i.e. average fraction of time that threads are executing) and average *IWT* (i.e. the time interval between the instant when a message is enqueued in the input queue until the instant when the message gets service from the processor). The dataset obtained as a result of the statistical simulation includes 792 samples for both MP protocols. Table 4.2 gives the descriptive statistics of the dataset using MP with ACK's protocol.

Table 4.2. Descriptive statistics of the MP with ACK's dataset

Statistics Name	Performance Measures				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
Mean	19.0801	0.2322	449.4143	0.4649	167.8480
Maximum	186.3973	0.8541	1027.3580	0.9509	356.9148
Minimum	0.0031	0.0007	20.6056	0.0119	2.1585
Standard Deviation	28.8380	0.2129	240.3182	0.2892	94.7545

Table 4.3 shows the statistical dataset obtained by using MP without ACK's protocol. Hybrid MP Dataset obtained by integrating the results for both MP with ACK's into MP with no ACK's. Table 4.4 gives descriptive statistics of a hybrid MP dataset.

Table 4.3. Descriptive statistics of the MP without ACK's dataset

Statistics Name	Performance Measures				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
Mean	12.76555	0.571891	280.7474	0.690725	133.531
Maximum	105.259	0.996528	687.25	0.995875	361.9174
Minimum	0.005515	0.065972	21.875	0.088186	0.125
Standard Deviation	16.04797	0.221036	150.7417	0.196795	79.48045

Table 4.4. Descriptive statistics of the Hybrid MP dataset

Statistics Name	Performance Measures				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
Mean	15.9108	0.401922	364.9264	0.57789	150.626
Maximum	186.3973	0.996528	1027.358	0.995875	361.9174
Minimum	0.003125	0.000729	20.60564	0.011857	0.125
Standard Deviation	23.53385	0.27562	217.4957	0.271731	89.08464

4.3. DSM Framework and Dataset Generation

Each SOME-Bus node can be represented by a set of queues through which messages of different types flow. Each node contains four major components: The processor handles all activities related to the scheduling of the threads. The arrival of data and ownership acknowledge messages causes threads to become ready for execution and therefore, affects the processor operation. The cache controller fills requests for data from the threads. The directory controller maintains the directory information for the portion of main memory that is located at its node and receives and processes data and ownership requests from the processor. The channel controller receives messages from the processor, cache or directory controllers and delivers them to the destination node. If the source and destination nodes of the message are different, the message is considered to be remote and is placed on the output queue associated with the output channel of the source node. When the channel becomes available, the message is transmitted and arrives at the input queue at the destination node. Messages that are broadcast or multicast arrive simultaneously at the destination input queues, else it is placed in local node.

Initially, a 4-node DSM-based SOME-Bus system is designed by using OPNET Modeler as shown in Figure 4.3. After testing the system and ensuring that it works correctly, it has been expanded to represent (16, 32 or 64) nodes. The

processor, cache controller, directory controller and channel controller are represented by queue modules with the symbols “pr”, “cac”, “dir” and “ch”, respectively. The function of the “hub” is to receive data and coherence messages from the channel module and send them to the other queue modules. The underlying process model that controls queue modules’ behavior is OPNET’s built-in acb_fifo model, which can be seen in Figure 4.2. OPNET’s built-in acb_fifo model has its own server and can concentrate multiple incoming packets streams into its single internal queuing resource. It also supports the First-in-First-out service ordering discipline and a way to control service times. The “init” state is used to initialize the process and setting the appropriate variables. If a packet arrives when the process is in “init” state, the process transitions to the “arrival” state, else it transitions to the “idle” state where it waits for packet arrival. The “arrival” state is used for receiving packets and starting service. In the “arrival” state, if the server is not busy then the process moves into the “svc_start” state, which in turn transitions to the “idle” state, where it waits either for packet arrival or service completion. While in the “idle” state, if the processing of a packet is completed, the process moves into the “svc_compl” state. While in the “svc_compl” state, if the queue is not empty, the process moves into the “svc_start” state.

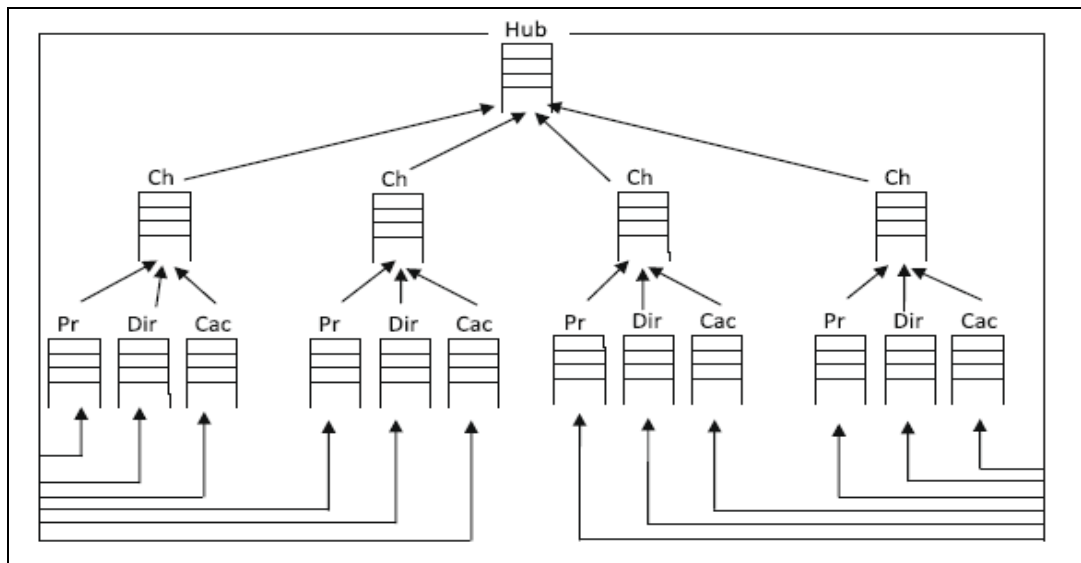


Figure 4.3. Node Model of a four-node DSM over SOME-Bus Architecture

The state of a cache block (i.e. cache line) is determined according to the MESI protocol. Each cache line is either Modified, i.e. the local cache has the only copy of the cached data in the system and it is dirty; Exclusive, i.e. only one cache has a copy of the block and it has not been modified; Shared, i.e. the local cache contains a valid, read-only copy of the data, and furthermore other caches may also have a read-only copy; or Invalid, i.e. the local cache does not have a valid copy of the data. Directory entries can be in the state Unowned, i.e. no cached copies in the system; Shared, i.e. zero or more read-only cached copies or Modified, i.e. one read-write cached copy in the system and the block may be in either dirty or (clean) exclusive state in the cache (Eisley et al., 2006). Each directory entry is associated with a bit vector (the copy set) that identifies the processors with a copy of the data block corresponding to that entry.

For a data request to a block in shared or unowned directory state, the block is supplied from the home node memory. The home node sends data acknowledge message with data to the requesting node. If the block is in exclusive directory state, the owner node is determined among remote nodes with uniform distribution by the home directory. In intervention forwarding and reply forwarding protocols, the home directory sends downgrade write back request message to the owner node, which has the modified block in its local cache. However, in strict request-response protocol, the home directory sends the address of the owner node to the requestor node. Then, the requester node's directory sends downgrade write back request message to the owner node. When the owner node's cache receives downgrade write back request message and if the protocol is reply forwarding or strict request-response, the owner node directly sends downgrade write back acknowledge message with data to the requesting directory and sends a revision message to the home directory. If the protocol is intervention forwarding, the owner node sends downgrade write back acknowledge message with data to the home directory.

For an ownership request to a block in unowned directory state, the home directory sends ownership acknowledge message with the requested block to the requesting node. The network transactions of an ownership request to a block in exclusive directory state is the same with the transactions of a data request to a block

in exclusive directory state. The only difference is the type of the messages. However, if the block is in shared state; in intervention forwarding protocol, the home directory sends invalidation messages to the sharer nodes and waits for acknowledgments from them. In reply forwarding protocol, the home directory initially sends the addresses of the sharers to the requestor node's directory and then sends invalidation messages to the sharer nodes. In strict request-response protocol, the home directory sends the addresses of the sharers to the requestor node's directory and then the requesting directory sends invalidation messages to the sharer nodes. When the sharer's cache receives an invalidation message, it sends the invalidation acknowledge message to the home directory if the protocol is intervention forwarding or it sends the invalidation acknowledge message to the requesting directory if the protocol is reply forwarding or strict-request response. In intervention forwarding protocol, when all invalidation acknowledge messages are received by the home directory, the home directory sends ownership acknowledge message with data if needed, to the requestor node. In reply forwarding and strict request-response protocol, when all invalidation acknowledge messages are received by the requestor directory, the home directory sends owner acknowledge message to local processor and data to local cache if needed, and then, it sends a revision message to the home directory.

Another type of message in the system is generated when a cache gets a block but has no empty space to put it in. At this time, the cache has to remove a random block, and notify the home directory of the block about this operation.

The traffic generation method used in this work is extensively described in Section 4.2. For making the experiments reproducible, the rest of the parameters used in this simulation must be described. Other major parameters of the simulation, which can be seen in Table 4.5, are the distribution of the thread run time (R), number of threads in each node chosen as 1 through 6, the fraction of write messages, the number of invalidation messages sent with every request for ownership message, the mean channel service time (T) for different types of packets, probability of a cache being full and probability of a block being in various states.

The processor at each node is assumed to be executing a program with several threads (selected from 1 to 6). In a real application execution, a large fraction of time will be spent by the processors doing calculations. At certain instants, these calculations need data in external memory and a remote memory access is performed. An important parameter in this respect is the computation to communication ratio, which tells us whether the execution of a certain application is dominated by useful computation, versus waiting for remote memory accesses (Heirman et al., 2007). In this simulation, the time between subsequent requests from the same node (called thread run time) has an exponential distribution with a mean of 100.

Table 4.5. System Parameters

Parameter	Value
Thread number in each node	Selected as 1, 2, 3, 4, 5 and 6
Mean thread run time	exponentially distributed with a mean of 100 clock cycles
Mean channel service time for a packet	varies between 5 and 100
Probability of write (ownership) request – $P(W)$	0.2 , 0.4 , 0.6
Probability of upgrade ownership request	0.2
Probability of a block being in modified state – $P(M)$	0.2 , 0.4 , 0.6 , 0.8
Probability of a block being in unowned state	0.1
Probability of the requestor being the only sharer	0.15
Owner node selection	Uniformly distributed
Probability of a cache being full	0.15
Sharer count	3
Nodes numbers	Selected as 16, 32 and 64

In the context of communication networks this time is also referred to as the think time, during which the processor or user ‘thinks’ about what request he will make next. The requesting message is an ownership request message with a probability of $P(W)$, or a data request message with a probability of $1 - P(W)$. $P(W)$ has the values 0.2, 0.4 and 0.6 whereas the probability that a block is found in modified state, $P(M)$, takes the values 0.2, 0.4, 0.6, and 0.8. These numbers are consistent with commonly observed memory reference patterns of real parallel applications and benchmarks. For instance, (Acacio et al., 2002) have experimented with five different parallel applications (i.e. EM3D, FFT, MP3D, Ocean and Unstructured) and they observed that write cycles constitute 25% to 68% of all memory cycles. In (Hu and John, 2006), the write miss percentage of the SPEC CPU INT 2000 benchmarks was reported to change from 13% to 52.74%. It was also reported in the same study that 20% to 55% of overall misses were to a modified cache block. The number of invalidation messages sent with every ownership request message is three. Table 4.6 shows the descriptive statistics obtained by the Opnet simulation modular using the DSM system.

Table 4.6. Descriptive statistics of the DSM dataset.

Statistics Name	Performance Measures				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
Mean	112.6242	0.445	578.9753	0.425801	234.7393
Maximum	718.8793	0.994489	1956.088	0.992933	1213.14
Minimum	0.003125	0.000729	2.799829	0.011857	0.329721
Standard Deviation	146.8254	0.310609	413.6286	0.258793	305.8993

The ratio T/R varies between 0.05 and 1. This range of the ratio is sufficient to capture the system behavior under most common configurations and cache behavior. Specifically, let m be the miss rate and F the number of instructions per

second performed by the processor at each node. In addition, let S be the mean message size in bytes and C the channel bandwidth (in bytes per second). Then, the ratio of the mean thread run time to the mean message transfer time $T/R = mSF/C$. In current high performance architectures, the ratio of F/C is in the range of $0.5 - 1$. For example, in Cray XT3 (Alam et al., 2008), $F = 4.8 \times 10^9$, and the links have a peak bandwidth of 7.6 GB/s. With small cache blocks and miss rate in the neighborhood of 10% or less (due to the fact that programmers are going to target and distribute their applications for maximum locality, thus most accesses on well behaved applications are going to fall in cache), the resulting ratio T/R is in the range of 0.05 to 1.

There are several applications for which upgrade misses account for an important fraction of the cache misses (Acacio et al., 2002). Upgrade misses are caused by a store instruction that finds a read-only copy of the data in the cache. For this kind of misses, the cache already has the valid data and only needs exclusive ownership. The directory must invalidate all the copies of the data but the one held by the requesting processor. The effect of upgrade misses is taken into account in the simulation by setting the probability of having an upgrade ownership request message to 0.2, which is consistent with the numbers given in (Acacio et al., 2002).

The value of the last parameter, the probability of a cache being full, is 15%. When a cache gets a block but has no empty space to put it in (full), it removes a random block and notifies the home directory of the block about this operation.

5. RESULTS AND DISCUSSION

Results were obtained by using four datasets. Based on the protocol of the programming model applied, the datasets represent: 1) MP with ACK's (includes 792 data points); 2) MP without ACK's (consists of 792 data points); 3) Hybrid MP (involves 1584 data points); and 4) DSM (contains 792 data points).

5.1. MFANN Prediction Model

The MFANN prediction model is shown in Figure 5.1. As is seen in Figure 5.1, the neural network structure contains two hidden layers. The first hidden layer has 9 neurons and the second hidden layer has 6 neurons. The network parameters have been optimized by try-and-error (i.e. after testing the neural network with several different configurations and observing that these numbers yield the lowest error rates for prediction) in order to reach the accurate results. A tansigmoid activation function is used in the hidden layers. A pure-linear activation function is used in the output layer.

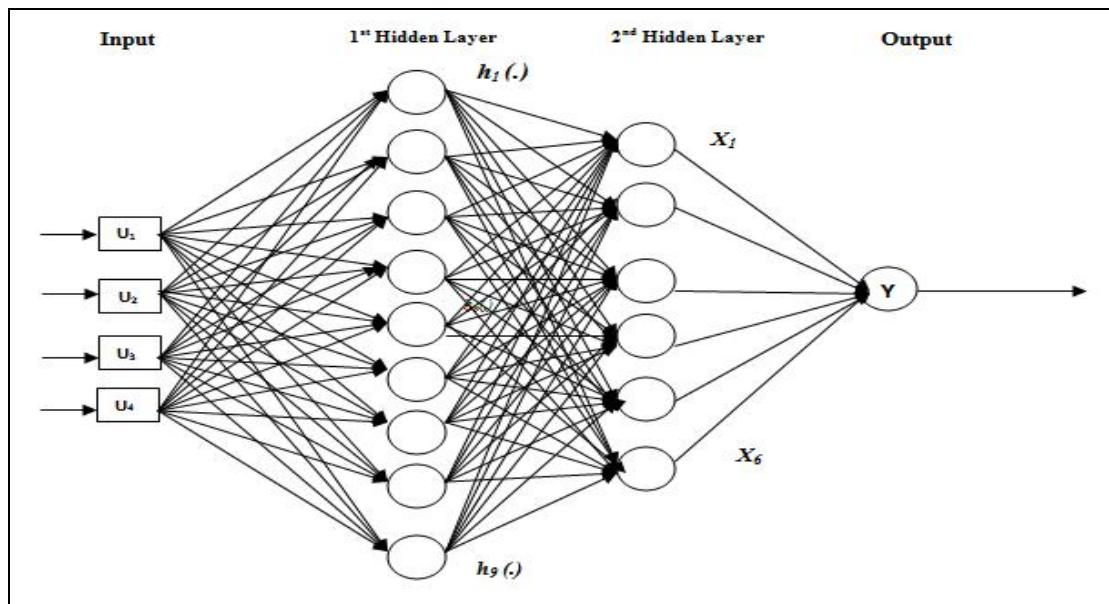


Figure 5.1. MFANN prediction model

The Levenberg -Marquardt (LM) algorithm is utilized for training the network. The other important parameters of the MFANN model are the number of epochs (selected as 500), the learning rate (selected as 0.02) and momentum (selected as 0.5). Parameters U_1 through U_4 represent the inputs, $h_1(.)$ through $h_9(.)$ and X_1 through X_6 represent the outputs of the first and second hidden layers, respectively, and Y is the output of the network.

5.2. SVR Prediction Model

It is well known that SVM generalization performance (estimation accuracy) depends on a good setting of hyper parameters C , e and the kernel parameters. The problem of optimal parameter selection is further complicated by the fact that SVM model complexity (and hence its generalization performance) depends on all three parameters. Recently, a practical method for selecting the value of C and e for SVM regression directly from the training data is proposed (Cherkassky and Ma, 2004). Specifically, the value of C is chosen as:

$$C = \max\left(\left|\bar{y} + 3s_y\right|, \left|\bar{y} - 3s_y\right|\right), \quad (5.1.)$$

where $\bar{y}$ is the mean of the training outputs and s_y is the standard deviation of the training outputs.

The value of e is selected as:

$$e(s, \mathbf{l}) = ts \sqrt{\frac{\ln \mathbf{l}}{\mathbf{l}}}, \quad (5.2.)$$

where s is the standard deviation of additive noise, $\mathbf{l}$ is the number of training samples and t is an empirically determined constant. (Cherkassky and Ma, 2004) suggests $t = 3$ for setting the value of e -insensitive zone. Hence, (5.2.) with $t = 3$ will be

$$e(s, \mathbf{l}) = 3s \sqrt{\frac{\ln \mathbf{l}}{\mathbf{l}}}, \quad (5.3.)$$

Note that using (5.3.) requires estimation of noise level s . This can be accomplished using standard noise estimation approaches:

$$(\hat{s})^2 = \frac{\mathbf{l}}{\mathbf{l} - d} \frac{1}{\mathbf{l}} \sum_{i=1}^{\mathbf{l}} (y_i - \hat{y}_i)^2, \quad (5.4.)$$

where $(y_i - \hat{y}_i)$ is the i th fitting error of the training data, d is the dimensionality of the input space and $\mathbf{l}$ is the number of training samples. Using the k -nearest neighbors method, the model complexity will be

$$d = \frac{\mathbf{l}}{k}, \quad (5.5.)$$

where k is the number of data points near the local estimated points. Combining (5.4.) and (5.5.), we obtain the following prescription for noise variance estimation via the k -nearest neighbors method:

$$(\hat{s})^2 = \frac{k}{k-1} \frac{1}{\mathbf{l}} \sum_{i=1}^{\mathbf{l}} (y_i - \hat{y}_i)^2 \quad (5.6.)$$

In general, the value of k varies between 2 and 6. Also, (Cherkassky and Ma, 2004) suggested setting $k=3$ and they tested it for different sample sizes and different noise levels. With $k=3$, (5.6.) becomes

$$(\hat{s})^2 = 1.5 \frac{1}{\mathbf{l}} \sum_{i=1}^{\mathbf{l}} (y_i - \hat{y}_i)^2 \quad (5.7.)$$

During the selection of the SVR model for performance measures prediction of the SOME-Bus multiprocessor, the following kernel functions are considered: linear and RBF. The optimal value of r for the RBF is determined by using cross validation. For the e -insensitive loss function, the study uses the mean and standard deviation of the training outputs in (5.2.) to calculate the regularization parameter C and the study uses (5.3.) to calculate e . The standard deviation of additive noise s is estimated directly from the training data using (5.7.).

5.3. Performance Measures

The performance of the prediction models are evaluated using R , SEE , MAE , RAE and $RRSE$ whose formulas are given in Eq. (5.8.) and Eq. (5.9.), Eq. (5.10.), Eq. (5.11.) and Eq. (5.12.), respectively (Haykin Simon, 1999; Witten and Frank, 2005)

$$R = \sqrt{1 - \frac{\sum_{i=1}^n (Y - Y')^2}{\sum_{i=1}^n (Y - \bar{Y})^2}} \quad (5.8.)$$

$$SEE = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y - Y')^2} \quad (5.9.)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y - Y'| \quad (5.10.)$$

$$RAE = \sqrt{\frac{\sum_{i=1}^n |Y - Y'|}{\sum_{i=1}^n |Y - \bar{Y}|}} \quad (5.11.)$$

$$RRSE = \sqrt{\frac{\sum_{i=1}^n (Y - Y')^2}{\sum_{i=1}^n (Y - \bar{Y})^2}} \quad (5.12.)$$

where n is the number of data points used for testing, Y is the observed value, Y' is the predicted value and $\bar{Y}$ is the average of the observed values.

5.4. Results and Discussion for MP with ACK's

Table 5.1 through 5.8 show the performance of all prediction models using different number of CV folds (10 up to 80). Based on the results the following general points can be made :

- § For all performance measures, the MFANN-based prediction model performs better (i.e., higher R and lower SEE , MAE , RAE and $RRSE$) than SVR-based, GRNN-based and MLR-based prediction models.
- § SVR-RBF model shows the second best performance for prediction.
- § The SEE for the MFANN-based prediction model decreases as the number of folds in the test set increases from 10 to 80. However, it is observed that the SEE of the ANN-based model increases as the number of folds exceeds 80.
- § The MFANN-based model performs a perfect job in predicting CU and PU (i.e., the SEE is almost tends to zero for both predictions). The prediction errors related to NRT and IWT are higher than the ones related to CWT . This is because of the high standard deviation of NRT and IWT in the dataset.
- § Although the MLR-based prediction model yields good performance for prediction of CU and PU , it does not show the same performance for prediction of CWT , NRT and IWT . This is because of the non linear characteristics of CWT , NRT and IWT .

- § Since there is no training phase in GRNN, the GRNN-based model produces results much faster than the MFANN-based and SVR-based prediction models.
- § The MFANN-based prediction model yields the lowest *SEE* for prediction of *PU*, where the *SEE* changes from 22.3406 to 14.2463.
- § MLR and SVR-L models show similar performance for prediction among all the CV folds.
- § The *R* values for prediction of *CWT*, *CU*, *NRT*, *PU*, and *IWT* are limit to 1 for all folds.
- § The training times for the MFANN-based models are much lower than that of SVR-based models.
- § The training phase for SVR-RBF model elapses long time to make the predictions compared against the ones obtained by other models. This is because of the usage of the Gridsearch algorithm in the SVR-RBF model to compute the optimum values of the related parameters.
- § The execution times for the SVR-RBF and SVR-L prediction models take time, whereas the execution times for MFANN, GRNN and MLR models are negligible (close to zero).
- § For *CWT*, MFANN is the best predictor and it has the lowest *SEE* (1.1782) using 80 folds CV and the highest *R* (0.9995) implementing 60 folds CV.
- § In *CWT*, excluding the linear models (SVR-L and MLR), increasing the number of CV folds relatively increases the prediction efficiency.
- § The linear models (SVR-L and MLR) represent the least significant tools to be used in measuring *CWT* and both models degrade in their performance when raising the CV folds.
- § For *CU*, the MFANN is the optimum technique to be used when predicting *CU* under a MP multiprocessor architecture and registers the highest *R* (0.9996) and lowest *SEE* (0.0054) using CV with 80 folds.
- § For *CU*, increasing the number of folds does not make a big difference in the values of performance measures.

- § For the *NRT*, the best results ($SEE = 14.2463$ and $R = 0.9979$) were obtained using the MFANN with 80 folds CV.
- § For *NRT*, considering SEE using 10 to 30 folds CV, the GRNN model performs better than the MFANN model.
- § For *PU*, and according to the results obtained the evaluators can be organized in descending order as: MFANN, GRNN, SVR-RBF, SVR-L and MLR.
- § For *PU*, relatively all the five predictors accurately share the same function minimizing the errors and boosting R 's.
- § Under *PU*, the highest R (0.9994) and the lowest SEE (0.0072) were obtained using folds 80 on the MFANN.
- § In *PU*, SVR-RBF model relatively shows typical values for R (0.9885) and SEE (0.0565) whatever CV changes from 10 to 80 folds.
- § For *IWT*, MFANN is the best predictor ($R = 0.9893$ using 70 folds CV and $SEE = 11.9345$ using 80 folds CV).
- § Assessing *IWT* based on SEE , GRNN model performs the best results using 10 to 50 folds CV.

5.5. Results and Discussion for MP without ACK's

Based on the results obtained in Table 5.9 through Table 5.16 the following comments can be made :

- § In general, prediction models for MP with no ACK's protocol perform better than the ones for both Hybrid MP and MP with ACK's.
- § The machine learning predicting evaluators can be ordered as: MFANN, GRNN, MLR, SVR-RBF and SVR-L.
- § MFANN records the highest results using 80 folds CV.
- § For the CWT, MFANN performs the highest values ($R = 0.9947$ and $SEE = 1.1835$) using 80 folds CV.
- § For CWT, based on SEE , GRNN technique gives the lowest values (1.0682) using 80 folds CV.

- § For CWT, the results prove that MLR is a well-competent compared to the robust machine learning techniques and it records high values ($R = 0.985302$ using 20 folds and $SEE=2.035674$ using 80 folds).
- § For CU, MFANN is the best predictor ($SEE = 0.0269$ and maximizes $R = 0.9906$ using 80 folds CV).
- § Assessing CU, MLR technique records higher findings than the other models including MFANN with 10 folds CV.
- § In summary: MFANN, MLR, SVR-RBF and GRNN models show relative typical results when assessing NRT.
- § For NRT, the lowest SEE (11.413) was obtained by using GRNN-based model over 80 folds CV.
- § Excluding MFANN, MLR-based model predicts measures for NRT better than GRNN, SVR-RBF and SVR-L.
- § MFANN is the best machine learning predictor evaluating PU using 80 folds CV.
- § Assessing PU, the performance tools can be ordered as: MFANN, MLR, GRNN, SVR-RBF and lastly SVR-L.
- § MLR and GRNN relatively show similar results estimating PU. Very often, MLR-based performs better than GRNN.
- § It is quite obvious that SVR-L is not a suitable technique to be used for predicting PU on MP without ACK's.
- § All the five machine learning techniques show good results in predicting the IWT, but MFANN is highly recommended because it gives the demanded results for R (0.9986).
- § The smallest R 's values for the (MFANN, MLR, SVR-RBF and GRNN) models is greater than or equal to 0.994.
- § IWT performance metric values prove the reliability and accuracy of the machine learning methods the thesis used.
- § The execution duration times for the training phase across MFANN and SVR were given in Tables 5.36, 5.37 and 5.38.

5.6. Results and Discussion for Hybrid MP

Table 5.17 through Table 5.24 show the performance of all prediction models for the hybrid MP case.

- § Hybrid MP prediction models perform better than the ones for MP with ACK's.
- § Increasing the CV fold numbers enhances the performance of all the machine learning models.
- § For *CWT*, the highest values were obtained using MFANN with 80 folds CV. The optimum values for *R* and *SEE* are 0.9941, 2.0764, respectively.
- § For *CWT* Sometimes, the linear models (MLR and SVR-L) sometimes perform better than the powerful ones (GRNN and SVR-RBF).
- § For *CWT*, in some situations, MLR and GRNN models share the same performance degree computing the correlation coefficient.
- § For *CU*, it is obvious that MFANN technique shows the best results for *R*. While considering out *SEE*, GRNN-based model gives the lowest value for the *SEE* (0.0188).
- § Following the numbers of the CV folds from 10 up to 50, GRNN-based model produces the best results in predicting *CU*.
- § SVR-L model is the less effective method to be used in evaluating CU over hybrid MP because it reveals low *R* (0.6347) and high *SEE* (0.2143).
- § MLR-based model is a robust predictor and intactly evaluates CU; moreover its results proved that MLR is a well-competent as well as the robust methods.
- § Considering *NRT*, using fold numbers for CV from 50 down to 10 the GRNN-based and MLR model perform better than even the MFANN model.
- § Focusing on the machine learning selected (MFANN, GRNN, MLR and SVR-RBF) models, their *R*'s tend to 1 and this fact indicates the close converge that all the four techniques are fairly competent and each can be used in predicting *NRT* in a multiprocessor system.

- § SVR-L is inconvenient method to be applied in assessing *NRT* over a MP multiprocessor network, because it gives high error rates.
- § In *PU*, considering *SEE*, GRNN-based model performs better than MLR and MFANN techniques.
- § Based on *PU*, generally, the values for *R* increase with respect to the increasing of CV fold number.
- § For *PU*, SVR-based models do a perfect job for calculating errors and correlation coefficients.
- § Predicting *IWT* and considering $R = 0.9877$, MFANN gives the best results. While considering $SEE = 8.0656$, GRNN-based model shows the best results.
- § GRNN, MLR and SVR-RBF lie in the same order calculating *R* and the weakest one (SVR-RBF using CV 70) shows *R* greater than or equal to $\{0.972\}$.
- § Execution duration times for the traing phase are showed in Tables 5.39 through Table 5.41.

5.7. Results and Discussion for DSM Results

Table 5.25 through Table 5.32 show the performance of all prediction models for the DSM case. Based on the results the following outlines made:

- § Machine learning methods can be organized descendingly based on their accuracy as: SVR-L, SVR-RBF, MLR, GRNN and MFANN.
- § Based on training and testing execution duration times, machine learning techniques can be ordered as: MFANN, SVR-RBF, SVR-L and MLR-based. Because of the non-existence of the training phase in GRNN-based, it performs the results faster than the other methods.
- § Considering the results accomplished using the CV method across all folds from 10 to 80 folds, the 80 CV fold usually shows the best results for MFANN, SVR-RBF and the MLR model.

- § MLR-based models relatively show similar results compared with the robust modern machine learning methods.
- § Predicting CWT, MFANN model performs the best results when implementing 80 folds CV, for example $R = 0.9969$ and $SEE = 0.0191$.
- § In CWT, the linear models (SVR-L and MLR) relatively report equal values for R , SEE , MAE , RAE and $RRSE$ over the whole CV from 80 down to 10 folds.
- § For CU, MFANN gives the best results for the correlation coefficient ($R = 0.9968$ using 70 folds CV) and errors ($SEE = 11.5186$; $MAE = 8.93$; $RAE = 0.27$; and $RRSE = 0.08$ using with 80 folds CV).
- § Assessing CU, the performance tools can be ordered as: MFANN, GRNN, MLR, SVR-RBF and SVR-L.
- § Predicting NRT, SVR-RBF shows the best results ($R = 0.998$ using 80 folds CV and $SEE = 30.217$, $MAE = 0.2$, $RAE = 0.60762$, $RRSE = 70\%$ using 80 folds CV).
- § In NRT the evaluators priority for the techniques can be listed in order as: SVR-RBF, MFANN, GRNN, MLR and SVR-L.
- § For NRT, the linear models (SVR-L and MLR) are not advisable to be used for predicting NRT for the DSM protocol.
- § Assessing PU, machine learning techniques can be ordered based on their accuracy as: MFANN, GRNN, SVR-RBF, MLR and SVR-L.
- § The best results for predicting PU were obtained using the MFANN model over DSM with 80 folds CV (i.e. $R = 0.9975$, $SEE = 33.0929$, $MAE = 26.38$, $RAE = 0.25$ and $RRSE = 0.07$).
- § In order to predict the IWT, GRNN, MFANN and SVR-RBF models are reliable to be used, whereas the linear (SVR-L and MLR) models fail to compete the robust methods.
- § Predicting IWT, the best results are obtained using the GRNN employing 80 folds CV ($R = 0.9924$, $SEE = 23.4818$, $MAE = 32.20$, $RAE = 0.31$ and $RRSE = 0.12$).

Table 5.1. Performance measures of the MP with ACK using 10-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9956	.9175	.7719	.7802	.9826	.5401	.6028	7.7719	9.4299	.5152	.5021	1.0502	1.9778	0.6129	.31	.2754	.4776	.9814	4.56	.07	.0933	.3848	.6337	7.40	2.19
CU	.9992	.9882	.8915	.9911	.9864	.0088	.0188	.0948	.0288	.0573	.0065	.0309	.1113	.0179	.0477	.1919	.3223	.407	.99	6.63	.0416	.1463	.4509	3.51	6.92
NRT	.9955	.993	.7689	.7712	.9893	2.3406	9.2179	51.3559	53.5735	5.1874	6.1704	7.9267	59.9359	19.6952	4.1953	.2840	.3091	.9278	9.20	1.97	.0943	.1171	.6357	3.81	4.62
PU	.9993	.9988	.9568	.9566	.9883	.0106	.0095	.0834	.085	.0566	.0078	.014	.1962	.0623	.0488	.1793	.1988	.0741	5.69	0.15	.0370	.0487	.2898	9.39	9.57
IWT	.9737	.9587	.8225	.8251	.9572	0.7994	3.8427	3.1487	3.6145	7.4746	2.5012	5.7576	8.5852	0.3444	5.6073	.3914	.41	.9387	8.96	8.94	.2219	.2738	.5664	6.54	8.98

Table 5.2. Performance measures of the MP with ACK using 20-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF					
CWT	.9961	.9236	.7449	.7794	.9925	.3610	.4976	7.4882	0.9333	.5404	.4282	0.5831	1.397	0.6251	.3723	.2720	.4688	.0514	3.64	.93	.0868	.3657	.6528	2.60	2.28
CU	.9992	.9922	.8869	.9926	.987	.0081	.0155	.0941	.0261	.0571	.0061	.0257	.1063	.0163	.0473	.1866	.2967	.3184	.10	6.41	.0393	.1227	.4561	2.25	6.84
NRT	.9964	.9925	.76	.7688	.9899	9.9550	9.2922	51.9981	55.632	4.2684	4.5583	7.8869	13.2991	25.0479	3.4044	.2703	.3118	.8969	1.87	1.58	.0848	.1191	.6432	4.69	4.24
PU	.9994	.9989	.9548	.9566	.9883	.0100	.009	.0827	.0851	.0566	.0073	.0132	.1935	.0622	.0488	.1750	.194	.9494	5.68	0.13	.0353	.0465	.2939	9.41	9.57
IWT	.9757	.9573	.8106	2.76	.9609	9.5336	3.616	3.4062	3.1712	6.234	2.4941	5.0851	6.235	0.5189	4.9452	.3943	.4066	.804	9.16	8.13	.2122	.2696	.5773	6.06	7.66

Table 5.3. Performance measures of the MP with ACK using 30-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE																
	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN
CWT	.9973	.9221	.7556	.7781	.9925	.9756	.5334	7.2198	0.9609	.5278	.2958	0.6251	1.9323	0.6206	.3666	.2557	.47	.2735	3.61	.90	.0731	.3652	.6345	2.66	2.23				
CU	.9994	.9923	.8827	.9931	.9866	.0074	.0154	.094	.0252	.0572	.0056	.0246	.0886	.016	.0474	.1794	.2985	.9997	.96	6.47	.0362	.1196	.462	1.84	6.88				
NRT	.9966	.9927	.7578	.7685	.9895	9.2222	8.7348	50.9108	55.6577	4.7925	4.3818	6.8753	11.04	25.0648	3.7212	.2696	.3095	.248	1.91	1.74	.0828	.1164	.6418	4.72	4.47				
PU	.9994	.9988	.9493	.9567	.9884	.0092	.009	.0831	.0849	.0566	.0070	.013	.165	.0621	.0487	.1728	.1938	.5699	5.63	0.13	.0336	.0465	.3021	9.36	9.56				
IWT	.9792	.9518	.7959	.8279	.96	7.7565	3.7838	2.6166	3.1414	6.5457	2.2585	5.3366	1.6074	0.4209	4.8154	.3883	.4135	.8606	9.06	7.98	.1927	.278	.5718	6.06	8.00				

Table 5.4. Performance measures of the MP with ACK using 40-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE												
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF		
CWT	.9975	.9344	.7074	.779	.992	.7644	.293	7.0384	0.9565	.659	.1664	.259	1.7394	0.6231	.3957	.2493	.457	.1817	3.62	.04	.0705	.3313	.6932	2.66	2.69
CU	.9994	.9929	.8713	.8977	.9869	.0068	.0144	.0939	.0941	.0572	.0053	.0228	.0909	.0758	.0474	.1757	.2862	.9571	2.37	6.50	.0339	.1126	.4688	4.19	6.85
NRT	.9969	.9931	.7462	.7692	.9896	8.3303	8.5584	50.6976	55.4807	4.556	3.9062	6.1737	19.4576	24.9542	3.2256	.2669	.3065	.0281	1.85	1.50	.0806	.1134	.6509	4.66	4.37
PU	.9995	.9988	.9515	.9566	.9884	.0090	.0087	.0812	.0851	.0565	.0069	.0127	.1441	.0622	.0486	.1724	.1926	.2452	5.70	0.09	.0331	.0462	.298	9.42	9.52
IWT	.9811	.9543	.7977	.8279	.9611	6.1882	3.8531	2.8934	3.1395	6.2092	1.1751	4.5218	1.8377	0.4695	4.4568	.3732	.4095	.8178	9.11	7.54	.1810	.2685	.5823	6.05	7.64

Table 5.5. Performance measures of the MP with ACK using 50-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9973	.9288	.68	.7788	.9924	.5529	.3918	7.131	0.9541	.5811	.0831	.2433	0.1775	0.6268	.3889	.2382	.4614	.1834	3.65	.10	.0684	.3343	.7568	2.65	2.35
CU	.9995	.9924	.8615	.8976	.9869	.0064	.0149	.0931	.0942	.0573	.0051	.0233	.105	.0758	.0474	.1758	.2918	.2326	2.39	6.48	.0334	.1155	.4847	4.22	6.90
NRT	.9974	.9917	.7191	.7688	.9896	.6480	8.691	50.6653	55.5459	4.7366	2.7539	6.01	17.4848	24.9847	3.5591	.2550	.309	.3004	1.88	1.66	.0726	.116	.6669	4.70	4.45
PU	.9995	.9986	.9429	.9571	.9885	.0083	.009	.0822	.0852	.0564	.0064	.0127	.128	.0686	.0486	.1668	.1981	.2169	8.35	0.09	.0309	.0488	.3087	9.46	9.52
IWT	.9836	.9451	.7929	.8279	.9609	4.5471	3.7239	1.7876	3.1319	6.249	0.3424	3.5524	8.5068	0.4619	4.3423	.3596	.4098	.7573	9.11	7.41	.1642	.2645	.5786	6.04	7.69

Table 5.6. Performance measures of the MP with ACK using 60-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF					
CWT	.9995	.924	.6057	.7791	.9927	.4062	.4779	6.313	0.9487	.4927	.0049	.2157	0.4713	0.6187	.3543	.1689	.4722	.1762	3.60	.84	.0313	.3512	.865	2.63	2.11
CU	.9995	.9928	.8745	.8976	.987	.0061	.0141	.0936	.0941	.0573	.0048	.0221	.0921	.0758	.0475	.1681	.2835	.9875	2.38	6.54	.0312	.1093	.4677	4.21	6.90
NRT	.9974	.9919	.6925	.7686	.99	6.3740	8.3469	48.9668	55.6017	4.0167	2.5806	5.5692	15.8411	25.0395	3.3954	.2544	.3101	.0772	1.90	1.58	.0739	.1164	.6767	4.71	4.15
PU	.9995	.9986	.9406	.9571	.9885	.0076	.009	.0814	.0852	.0565	.0060	.0128	.1203	.0687	.0488	.1614	.2007	.0754	8.36	0.16	.0293	.0492	.3123	9.45	9.55
IWT	.9851	.9443	.7769	.8281	.9616	3.6680	4.0401	2.0667	3.1063	6.0344	.7557	4.0185	7.2787	0.4028	4.5313	.3506	.4211	.0538	9.04	7.64	.1561	.2751	.5878	6.02	7.46

Table 5.7. Performance measures of the MP with ACK using 70-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF											
CWT	.9981	.9298	.6057	.7788	.9921	.3308	.4252	6.313	0.9572	.5788	.9522	.5908	0.4713	0.62	.3558	.2306	.4629	.1762	3.61	.88	.0613	.3253	.865	2.66	2.66
	.9995	.9916	.8214	.8975	.987	.0057	.0142	.0924	.0942	.0571	.0044	.0218	.088	.0759	.0472	.1668	.2927	.0569	2.39	6.38	.0312	.117	.5162	4.21	6.82
	.9979	.99	.7137	.7693	.99	4.8896	8.6448	49.8528	55.4193	3.9499	1.5577	5.6662	11.0137	24.8918	3.1311	.2483	.3176	.2408	1.83	1.45	.0691	.1224	.6751	4.64	4.12
PU	.9995	.9986	.9411	.957	.9885	.0075	.0089	.0806	.0853	.0566	.0059	.0128	.1078	.0687	.0488	.1613	.1972	.9694	8.37	0.17	.0289	.0488	.3097	9.48	9.56
IWT	.9893	.9472	.7298	.8281	.9619	2.5396	3.3184	1.7245	3.1109	5.9861	.1736	2.1546	8.2555	0.4132	4.2781	.3414	.4103	.953	9.06	7.33	.1427	.2595	.6447	6.04	7.42

Table 5.8. Performance measures of the MP with ACK using 80-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF											
CWT	.9980	.9285	.6198	.7791	.992	.1782	.3606	6.3505	0.9464	.5854	.8832	.5561	2.9327	0.6181	.3987	.2266	.4671	.2057	3.60	.95	.0578	.3261	.9113	2.64	2.70
	.9996	.9898	.8578	.9938	.9868	.0054	.0143	.091	.0238	.0573	.0044	.022	.0764	.0153	.0474	.1631	.2955	.0067	.57	6.48	.0296	.1203	.4853	1.19	6.89
	.9979	.9913	.6367	.7678	.9894	4.2463	8.0109	47.0421	54.4348	4.8452	1.2606	4.2023	17.1418	20.6416	3.5372	.2457	.3199	.2584	9.71	1.65	.0663	.1195	.7199	4.22	4.49
PU	.9994	.9984	.9103	.894	.9885	.0072	.0089	.0811	.0956	.0565	.0056	.0125	.1087	.0746	.0487	.1679	.2006	.1675	1.70	0.12	.0410	.0505	.4497	4.88	9.53
IWT	.9867	.9578	.757	.8278	.9618	1.9345	3.7176	1.0856	3.1457	5.9849	.8537	4.9858	1.9555	0.4651	4.5395	.3326	.4236	.8935	9.10	7.64	.1421	.2876	.5986	6.05	7.40

Table 5.9. Performance measures of the MP without ACK using 10-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN														
CWT	.9895	.9824	.983856	.5921	.9814	.3254	.2266	.768078	3.4221	.0843	.5950	.8341	.181793	.6463	.5302	.3835	.3328	.328012	9.91	3.99	.1483	.1775	.174335	3.60	9.21
CU	.9760	.9813	.979028	.6846	.9265	.0478	.0262	.0441	.1613	.0906	.0338	.0413	.027645	.1272	.078	.4273	.3756	.385869	8.48	2.02	.2177	.1889	.201356	2.96	0.96
NRT	.9805	.9827	.985083	.6535	.977	6.9111	2.5527	4.4547416	.7595	2.4283	8.8872	6.1724	1.90433	3.8138	5.4917	.3880	.316	.306582	3.77	2.18	.1944	.1767	.164657	6.87	1.35
PU	.9858	.9877	.988062	.4757	.9026	.0325	.014	.029009	.1734	.0884	.0231	.0291	.014197	.1387	.076	.3777	.2931	.293176	4.90	6.52	.1665	.1477	.146856	7.95	4.84
IWT	.9962	.996	.995731	.8308	.9934	.9194	.0488	.748863	5.4836	.1593	.6287	.8451	.977185	3.1778	.1468	.2622	.2101	.208524	8.93	.59	.0879	.0859	.085743	7.23	1.52

Table 5.10. Performance measures of the MP without ACK using 20-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9897	.984	.985302	.5912	.9833	.9874	.1011	.524081	3.295	.9232	.3977	.5805	.10708	.5979	.4548	.3666	.3201	.318457	9.46	3.30	.1427	.1689	.164946	2.83	8.21
CU	.9823	.9806	.979919	.6849	.9321	.0410	.0251	.04231	.1612	.0878	.0304	.0412	.026084	.1271	.0764	.4073	.3715	.378543	8.40	1.13	.1884	.191	.195326	2.88	9.71
NRT	.9864	.9835	.982534	.642	.9755	2.1538	1.9476	5.11504	16.4984	3.2515	6.3223	4.5004	1.90375	3.4113	6.0456	.3616	.3041	.305478	3.47	2.62	.1649	.1633	.169686	7.23	2.04
PU	.9874	.9889	.989307	.4757	.9095	.0305	.0139	.027114	.1734	.0857	.0222	.0273	.013731	.1387	.0741	.3729	.2937	.289316	4.97	5.41	.1582	.1408	.13863	8.01	3.47
IWT	.9966	.9962	.99589	.8302	.9944	.3294	.8696	.293586	5.5257	.4261	.3491	.4091	.951093	3.2741	.5471	.2534	.206	.205452	9.07	.71	.0814	.0826	.080993	7.28	0.60

Table 5.14 Performance measures of the MP without ACK using 60-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF											
CWT	.9933	.9849	.971233	.5912	.9852	.3714	.0718	.151064	3.2997	.7526	.0746	.1915	.088986	.5963	.3331	.3330	.311	.312983	9.43	2.18	.1160	.1514	.161484	2.82	7.14
	.9884	.9756	.978067	.6852	.9353	.0314	.0242	.038093	.1612	.0861	.0246	.0387	.024312	.127	.0753	.3768	.3761	.367322	8.36	0.51	.1552	.1983	.188141	2.88	8.92
NRT	.9921	.9786	.979445	.6756	.9823	6.6227	1.8179	1.78178	12.7069	8.4147	3.2648	2.8973	1.54653	0.098	3.1934	.3248	.3094	.301608	0.87	0.36	.1254	.1658	.158039	4.71	8.69
	.9924	.9877	.988717	.4801	.9112	.0219	.0134	.023873	.1729	.0849	.0165	.0233	.01334	.1384	.0734	.3281	.2866	.288124	4.82	5.01	.1237	.1294	.130726	7.84	3.14
IWT	.9982	.9947	.996159	.8301	.995	.4113	.8871	.637376	5.5394	.962	.3135	.7349	.892794	3.2604	.4146	.2248	.2037	.203376	9.02	.51	.0598	.0787	.074288	7.27	0.01

Table 5.15. Performance measures of the MP without ACK using 70-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN														
CWT	.9943	.9805	.985701	.5925	.9791	.2510	.1463	.08117	3.2864	.2743	.9603	.2614	.049647	.588	.4458	.3131	.3193	.304161	9.37	3.22	.1059	.1647	.146087	2.76	0.39
	.9899	.978	.979444	.6854	.9352	.0292	.0241	.036059	.1611	.0862	.0229	.0375	.023342	.1269	.0753	.3621	.3712	.364982	8.33	0.53	.1465	.1893	.182337	2.85	8.99
NRT	.9921	.9805	.98096	.7121	.9871	4.7510	1.6744	1.65176	10.728	4.3118	2.4384	1.0154	1.70173	8.3291	1.9352	.3244	.3047	.302213	9.41	.38	.1234	.1516	.153811	2.87	6.00
PU	.9933	.9886	.985972	.4787	.9096	.0203	.0132	.023734	.1731	.0853	.0156	.023	.014262	.1385	.0739	.3207	.2842	.295204	4.84	5.29	.1160	.1272	.136115	7.90	3.33
IWT	.9985	.996	.995692	.83	.9952	.0698	.7667	.240802	5.5433	.7898	.1472	.2912	.73353	3.2564	.3751	.2217	.1977	.202885	9.03	.45	.0558	.0708	.074644	7.29	.80

Table 5.16. Performance measures of the MP without ACK using 80-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																
	MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN														
CWT	.9947	.9845	.979339	.5914	.9846	.1835	.0682	.035674	3.2959	.8066	.9279	.0137	.066999	.6001	.3592	.3102	.3032	.315387	9.46	2.42	.1039	.1451	.158522	2.81	7.48
CU	.9906	.9771	.978341	.9354	.6851	.0269	.0243	.036664	.1612	.087	.0210	.0369	.023859	.127	.0751	.3464	.3742	.367251	8.38	0.38	.1381	.1894	.18489	2.88	8.57
NRT	.9941	.9815	.982384	.6654	.984	2.7283	1.413	0.28162	16.5161	3.0275	1.5065	0.6339	1.23996	2.3316	1.0474	.3125	.2973	.300444	2.64	.25	.1103	.1474	.149754	7.25	6.78
PU	.9941	.9841	.986175	.4784	.9107	.0194	.0137	.022631	.1731	.085	.0150	.0237	.013375	.1385	.0736	.3115	.2987	.288037	4.83	5.08	.1108	.1422	.13114	7.90	3.18
IWT	.9986	.9943	.99611	.83	.9951	.6524	.8044	.122179	5.5402	.874	.8143	.2553	.756611	3.2477	.4549	.2106	.201	.197241	9.01	.57	.0511	.076	.070808	7.27	.90

Table 5.17. Performance measures of the Hybrid MP using 10-fold CV

Performance Measures	<i>R</i>		<i>SEE</i>		<i>MAE</i>		<i>RAE</i>		<i>RRSE</i>	
	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN
<i>CWT</i>	.9901	.9342 .934182 .8661	.9757	.2250 .9634 .323037 7.97123	.1614	.9983 .3230 .9634340 .23536	.0593	.3639 .4365 .436502 .674701	.134542	.1416 .3521 .352131 .809338
<i>CU</i>	.9883	.9902 .976319 .6346	.9667	.0416 .0231 8.87822 .2143	.0885	.0277 .0380 .444116 .1747	.0782	.3448 .3149 .333374 .745931	.334011	.1519 .1385 .212731 .777323
<i>NRT</i>	.9876	.9908 .991601 .8616	.9757	5.9922 6.41847 .9425550 .7746	7.7802	4.0582 8.76666 6.0771528 .1481	6.407	.3659 .3022 .299672 .713734	.147068	.1566 .1329 .128803 .716886
<i>PU</i>	.9925	.9960 .996286 .8792	.9621	.0332 .0120 .022617 .2006	.0863	.0224 .0238 .011857 .1553	.0754	.3165 .2314 .23112 .692115	.336204	.1227 .0879 .084062 .737731
<i>IWT</i>	.9796	.9745 .976216 .80865	.9727	7.8887 .5593 8.89031 9.19194	0.7422	1.1415 9.0941 .3754177 9.1644	0.4272	.3829 .3347 .332389 .647116	.137043	.2019 .2160 .213882 .688108

Table 5.18. Performance measures of the Hybrid MP using 20-fold CV

Performance Measures	<i>R</i>		<i>SEE</i>		<i>MAE</i>		<i>RAE</i>		<i>RRSE</i>	
	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN
<i>CWT</i>	.9907	.9390 .939038 .8656	.97651	.0044 .8501 .865004 7.2444	.1616	.9372 .8650 .850126 .8278	.0431	.3597 .4267 .426691 .642158	.134	.1357 .3315 .331539 .817272
<i>CU</i>	.9899	.9916 .212731 .6347	.9694	.0385 .0205 7.86036 .2143	.0865	.0261 .0345 .189175 .1747	.0771	.3348 .2978 .327562 .745845	.329292	.1413 .1268 .203186 .777262
<i>NRT</i>	.9881	.9918 .99208 .8616	.9816	3.6278 5.1968 6.33063 14.7132	1.6033	4.1294 6.2719 5.2526 00.5928	3.1504	.3692 .2931 .292424 .746609	.128935	.1539 .1231 .122215 .704316
<i>PU</i>	.9934	.9960 .996367 .8799	.9642	.0301 .0116 .021685 .2004	.0842	.0212 .0228 .011381 .1551	.0742	.3093 .2276 .226418 .691442	.330618	.1134 .0849 .081308 .736997
<i>IWT</i>	.9819	.9753 .977596 .86785	.972	6.6464 .3583 7.96089 9.59845	1.01	1.2506 8.7909 .166472 1.8258	0.4066	.3858 .3312 .327353 .720058	.136765	.1887 .2138 .202307 .713953

Table 5.19. Performance measures of the Hybrid MP using 30-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE																	
	MLR	SVR-L	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF												
CWT	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF											
	.9918	.9366	.936627	.8654	.9805	.7956	.8146	.802899	7.251	.6429	.9112	.8029	.814575	.8342	.9546	.4265	.42652	.642363	.127671	.1269	.3367	.336677	.817398	.197138		
	.9912	.9921	.991872	.6342	.9704	.0354	.0197	.033483	.2144	.0854	.0243	.0333	.020097	.1748	.0764	.3247	.2914	.294143	.746098	.32633	.1311	.1226	.123521	.777613	.309732	
NRT	.9896	.9920	.991466	.8616	.984	0.7371	5.3072	6.74188	14.7123	8.7665	2.6594	5.8753	5.44492	28.511	2.2661	.3583	.2928	.294617	.007468	.12401	.1447	.1209	.125363	.523872	.178175	
	PU	.9939	.9963	.996454	.88	.9652	.0291	.0114	.021119	.2003	.0833	.0207	.0212	.011393	.1552	.0735	.3058	.2254	.225793	.691687	.327473	.1098	.0797	.078935	.736795	.306299
		IWT	.9832	.9751	.97549	.86785	.972	5.5021	.2311	8.41668	9.5467	1.0183	0.6457	7.9290	.274977	0.2341	0.1071	.3761	.3287	.329753	.007231	.132774	.1776	.2062	.209697	.007138

Table 5.20. Performance measures of the Hybrid MP using 40-fold CV

Performance Measures	R		SEE		MAE		RAE		RRSE												
	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN	MLR	SVR-L SVR-RBF MFANN GRNN											
CWT	.9922	.9470 .94696 .8651 .9858	.5882	.7040 .059025 7.2581	.6846	.8056	.0590	.704028	.8339	.9182	.3503	.4115	.411489	.642473	.121852	.1249	.3017	.301734	.817839	.200177	
CU	.9915	.9919 .991715 .6338 .9704	.0350	.0199 .033509	.2145	.0854	.0241	.0336	.019852	.1748	.0764	.3234	.2926	.291592	.746145	.32633	.1299	.1238	.123874	.777955	.309755
NRT	.9908	.9918 .991277 .8616 .9849	8.6552	5.2397 6.06066 14.7121	6.6764	1.7274	5.9047	5.18964	28.4789	1.0876	.3510	.2939	.292408	.006448	.117415	.1359	.1221	.123594	.523872	.173143	
PU	.9944	.9963 .996626 .8797 .965	.0271	.0113 .020156	.2004	.0834	.0195	.0206	.011091	.1553	.0735	.2997	.2235	.224416	.691722	.327609	.1038	.0769	.076523	.737033	.306635
IWT	.9846	.9761 .975957 .86785 .9715	5.2387	.0723 7.37953 9.781	1.1933	0.5767	7.2416	.175433	9.7498	0.1547	.3737	.3237	.328085	.007298	.13341	.1738	.1984	.201102	.007146	.237808	

Table 5.22. Performance measures of the Hybrid MP using 60-fold CV

Performance Measures	R		SSE		MAE		RAE		RMSE																
	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF											
CWT	.9936	.9483	.948258	.8654	.8412	.2856	.7046	.864035	7.2561	5.8356	.6463	.8640	.704563	.8356	.8776	.3374	.4109	.410932	.642652	.45049	.1140	.2906	.290634	.817819	.648215
	.9927	.9919	.992115	.6335	.972	.0319	.0195	.031771	21.46	.0842	.0228	.0320	.019101	.1749	.0759	.3153	.2894	.287849	.746709	.323964	.1196	.1196	.119284	.778434	.30525
	.9919	.9918	.991092	.859387	.9849	.68759	5.0491	5.34563	14.8651	4.765	1.0193	5.2678	5.10951	28.987	8.8723	.3457	.2915	.292332	.007423	.11321	.1281	.1199	.12221	.523872	.17347
PU	.9950	.9962	.99594	.8616	.9677	.0260	.0112	.020011	.2002	.08329	.0193	.0196	.011271	.1551	.0754	.2968	.2267	.226766	.691439	.3277	.0994	.0764	.07888	.736601	.30641
IWT	.9863	.9755	.974791	.86687	.9734	3.8407	.0981	7.13555	9.5632	0.4826	.9830	6.8959	.163812	0.3241	.8688	.3641	.3251	.326494	.007344	.129674	.1596	.1956	.1986	.007197	.229856

Table 5.23. Performance measures of the Hybrid MP using 70-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9927	.9410	.940965	.8653	.842	.2389	.7868	.956717	7.2578	5.2676	.6308	.9567	.786835	.8375	.8933
<i>CU</i>	.9933	.9922	.991795	.6337	.977	.0310	.0191	.031399	.2145	.0844	.0221	.0313	.019011	.1748	.07972
<i>NRT</i>	.9924	.9909	.990221	.8556	.9881	6.1904	5.2481	5.67884	14.7165	2.8123	0.1116	5.3851	5.27368	28.6541	8.431
<i>PU</i>	.9956	.9957	.996564	.8803	.9687	.0240	.0112	.019182	.2003	.08763	.0181	.0198	.011055	.1552	.0767
<i>IWT</i>	.9868	.9707	.973124	.86788	.972	3.3627	.1583	6.83955	9.4476	1.0064	.5867	6.6094	28.1819	7.1871	0.0379
												.3312	.329775	.007321	.131881
												.1584	.2024	.199686	.00721

Table 5.24. Performance measures of the Hybrid MP using 80-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9941	.9482	.948237	.8655	.842	.0764	.7905	.628732	7.2571	5.26	.5370	.6287	.790513	.8344	.8957
<i>CU</i>	.9935	.9924	.992062	.6337	.9787	.0297	.0188	.031018	.2146	.0851	.0215	.0310	.01887	.1748	.07732
<i>NRT</i>	.9928	.9909	.991353	.8612	.9878	5.3455	4.7984	4.51233	14.892	0.7612	9.4606	4.2635	4.96125	28.9432	7.4521
<i>PU</i>	.9962	.9960	.996296	.8801	.9723	.0225	.0113	.019375	.2003	.08872	.0169	.0197	.011146	.1552	.0768
<i>IWT</i>	.9877	.9748	.97465	.8688	.9732	2.8292	.0656	6.80337	9.5652	0.5365	.4490	6.3017	.075214	7.188	.8437
												.3218	.32534	.007378	.12934
												.1503	.1908	.197691	.007276

Table 5.25. Performance measures of the DSM using 10-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE												
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9858	.8728	.7172	.7132	.86	.0387	.055	.1661	.1679	.1225	.03	.11	.13	.13	.09	.37	.53	.68	.67	.46	.16	.48	.70	.70	.51
CU	.9950	.9887	.8132	.6075	.4166	5.7076	6.1614	1.5623	31.676	52.88	0.66	3.21	2.86	07.31	28.72	.28	.34	.53	.79	.94	.10	.15	.58	.84	.97
NRT	.9799	.9315	.8063	.8056	.9967	.0418	.0407	.1287	.1289	3.6226	.03	.08	.10	.10	.22	.40	.48	.54	.55	.56	.20	.35	.59	.59	.81
PU	.9958	.986	.59	.4325	.96	5.0462	6.5581	69.845	58.386	.0761	2.62	2.72	16.92	86.31	.06	.27	.36	.50	.89	.30	.09	.17	.54	.91	.29
IWT	.9237	.9349	.3725	.2347	.638	49.3403	7.7129	81.629	10.267	16.832	7.08	2.09	69.49	81.71	1.83	.49	.53	.76	.81	.28	.31	.35	.93	.93	.96

Table 5.26. Performance measures of the DSM using 20-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE												
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9853	.8578	.7175	.7138	.8632	.0381	.0567	.166	.1677	.1217	.03	.12	.13	.13	.09	.37	.54	.68	.67	.46	.16	.50	.70	.70	.51
CU	.9955	.9894	.8136	.6236	.4166	4.9655	5.1327	1.4652	30.305	52.546	1.02	2.20	2.79	06.21	28.50	.28	.33	.53	.78	.94	.10	.14	.58	.83	.97
NRT	.9842	.9291	.8064	.8061	.9969	.0368	.0401	.1287	.1288	2.4673	.03	.08	.09	.10	.21	.38	.48	.54	.54	.06	.17	.35	.59	.59	.78
PU	.9963	.9882	.843	.4415	.9605	3.1084	1.9937	69.266	56.218	.0752	1.62	4.59	16.31	84.49	.06	.27	.35	.50	.88	9.39	.09	.15	.54	.91	9.06
IWT	.9524	.9433	.3719	.2397	.648	3.5957	4.6158	81.737	09.763	13.493	3.30	9.85	69.67	81.30	9.32	.52	.51	.76	.81	.26	.30	.33	.93	.99	.97

Table 5.27. Performance measures of the DSM using 30-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9904	.8758	.7172	.7136	.8647	.0334	.0534	.1661	.1678	.1212	.02	.11	.13	.13	.09
<i>CU</i>	.9961	.9912	.8135	.6249	.4299	3.6850	3.9845	1.4818	29.998	52.181	0.09	0.26	2.79	05.88	28.22
<i>NRT</i>	.9840	.9334	.8061	.8059	.997	.0365	.0391	.1287	.1288	2.076	.03	.07	.10	.10	.20
<i>PU</i>	.9967	.9893	.8428	.443	.9607	0.3773	7.7343	69.373	56.026	.0742	0.29	9.88	16.25	84.53	.06
<i>IWT</i>	.9679	.9783	.3715	.2411	.648	0.4093	5.9411	81.796	09.64	11.554	5.36	7.53	69.77	81.17	8.52

Table 5.28. Performance measures of the DSM using 40-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9901	.8785	.7175	.7143	.8662	.0322	.0534	.166	.1676	.1205	.02	.11	.13	.13	.09
<i>CU</i>	.9962	.9919	.8139	.6309	.4284	3.5338	3.3068	1.3909	29.538	52.175	0.01	9.10	2.73	05.50	28.24
<i>NRT</i>	.9897	.9325	.8056	.8052	.997	.0304	.0392	.1289	.1291	2.0172	.02	.07	.10	.10	.21
<i>PU</i>	.9971	.9871	.843	.448	.9593	7.0657	0.0121	69.256	55.193	.0763	8.17	3.64	16.15	83.58	.06
<i>IWT</i>	.9641	.9744	.3739	.2425	.656	1.7304	1.0789	81.382	09.502	09.587	7.05	3.45	69.56	81.03	7.33

Table 5.29. Performance measures of the DSM using 50-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9902	.8706	.7174	.7139	.8659	.0310	.0335	.166	.1677	.1207	.02	.11	.13	.13	.09
<i>CU</i>	.9961	.9913	.8138	.6286	.4289	3.5201	3.3861	1.4242	29.684	52.135	0.28	8.81	2.74	05.67	28.23
<i>NRI</i>	.9877	.9387	.8062	.8057	.997	.0317	.0371	.1287	.1289	1.7172	.02	.06	.10	.10	.21
<i>PU</i>	.9966	.988	.843	.4448	.9607	9.2795	8.5155	69.253	55.688	.0739	9.23	0.28	16.13	84.04	.06
<i>IWT</i>	.9684	.9795	.3736	.2419	.656	7.8885	6.0365	81.443	09.526	09.587	3.89	9.42	69.73	80.93	7.33

Table 5.30. Performance measures of the DSM using 60-fold CV

Performance Measures	R			SEE			MAE			RAE			RRSE		
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF
<i>CWT</i>	.9899	.8898	.7172	.7134	.866	.0306	.0384	.1661	.1678	.1207	.02	.10	.13	.13	.09
<i>CU</i>	.9960	.9919	.8135	.6286	.4287	3.0419	3.1865	1.4885	29.712	52.106	.94	8.54	2.79	05.71	28.22
<i>NRI</i>	.9863	.9572	.8058	.805	.996	.0309	.0463	.1288	.1291	1.5172	.02	.07	.10	.10	.20
<i>PU</i>	.9972	.9887	.8426	.4438	.9707	6.5232	6.7467	69.597	55.881	.0739	7.78	7.16	16.39	84.62	.06
<i>IWT</i>	.9566	.9844	.3726	.242	.696	9.0057	4.2996	81.612	09.488	07.341	5.16	9.62	69.76	80.93	6.93

Table 5.31. Performance measures of the DSM using 70-fold CV

Performance Measures	R			SEE			MAE			R4E			RRSE												
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9907	.8743	.7174	.7136	.867	.0293	.0521	.166	.1677	.1202	.02	.10	.13	.13	.09	.35	.52	.68	.67	.45	.14	.44	.70	.70	.50
CU	.9968	.9916	.8137	.6297	.4291	1.5194	3.0934	1.4421	29.592	52.103	.93	8.01	2.76	05.54	28.23	.27	.32	.53	.77	.94	.08	.12	.58	.82	.97
NRI	.9881	.916	.806	.8052	.998	.0285	.0386	.1288	.129	0.6172	.02	.07	.10	.10	.20	.36	.47	.54	.55	.6256	.15	.32	.59	.59	.71
PU	.9972	.988	.8429	.4461	.9751	4.4136	7.194	69.353	55.648	.0721	6.87	5.39	16.21	84.32	.06	.26	.34	.50	.88	.29	.08	.14	.54	.91	.28
IWT	.9588	.9852	.3737	.2436	.716	6.0201	0.22	81.413	09.364	06.942	3.19	8.70	69.66	80.73	5.43	.48	.38	.76	.81	.21	.26	.17	.93	.99	.99

Table 5.32. Performance measures of the DSM using 80-fold CV

Performance Measures	R			SEE			MAE			R4E			RRSE												
	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF	MFANN	GRNN	MLR	SVR-L	SVR-RBF										
CWT	.9969	.8748	.7174	.7135	.8672	.0191	.0458	.166	.1678	.1201	.01	.11	.13	.13	.09	.29	.52	.68	.67	.45	.08	.48	.70	.70	.50
CU	.9965	.9918	.8137	.6311	.4347	1.5186	2.9779	1.4562	29.494	51.981	.93	7.84	2.76	05.51	28.15	.27	.32	.53	.77	.94	.08	.12	.58	.82	.97
NRI	.9886	.9646	.806	.8052	.998	.0283	.0359	.1288	.129	0.2172	.02	.07	.10	.10	.20	.36	.42	.54	.55	.6076	.15	.26	.59	.59	.70
PU	.9975	.9883	.8427	.4469	.9872	3.0929	8.3739	69.483	55.431	.0716	6.38	6.63	16.29	83.85	.06	.25	.34	.50	.88	.27	.07	.14	.54	.91	.28
IWT	.9704	.9924	.3739	.2437	.851	3.5489	3.4818	81.389	09.344	03.098	1.60	2.20	69.60	80.79	3.81	.47	.31	.76	.81	.21	.25	.12	.93	.99	.99

Table 5.33. Training times for the MFANN on MP with ACK's using different number of folds

Fold no.	Training Time(s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	1.26	1.170	1.180	1.150	1.200
20	1.14	1.320	1.160	1.350	1.190
30	1.28	1.170	1.170	1.170	1.200
40	1.15	1.180	1.360	1.160	1.200
50	1.17	1.340	1.200	1.150	1.200
60	1.15	1.17	1.35	1.18	1.18
70	1.16	1.32	1.37	1.2	1.4
80	1.17	1.41	1.38	1.39	1.17

Table 5.34. Training times for the SVR-L on MP with ACK's using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	1724	73.95	201.2	96.53	175.2
20	1579	75.74	202.3	98.09	176.8
30	1607	75.81	204.1	98.42	176.7
40	1642	76.13	203.6	97.97	176.3
50	1693	75.5	203.3	98.08	176.5
60	1572	75.67	203.1	98.53	176.8
70	1658	75.88	203.8	98.8	175.6
80	1761	75.78	203.2	98.75	176.6

Table 5.35. Training times for the SVR-RBF on MP with ACK's using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	3012	317.8	317.0	386.1	787.8
20	3012	317.1	315.7	386.4	786.6
30	3012	317.9	315.2	386.6	785.9
40	3012	318.7	315.3	386.5	785.8
50	3012	318.6	315.9	386.8	786.5
60	3012	319.4	315.4	386.8	790.4
70	3013	319.9	316.9	387.9	788.6
80	3017	319.9	315.9	388.7	784.6

Table 5.36. Training times for the MFANN on MP without ACK's using different number of folds

Fold no.	Training Time(s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	1.15	1.18	1.17	1.16	1.18
20	1.15	1.15	1.45	1.18	1.17
30	1.47	1.18	1.16	1.16	1.15
40	1.45	1.15	1.16	1.16	1.17
50	1.46	1.47	1.14	1.15	1.18
60	1.14	1.15	1.14	1.16	1.16
70	1.16	1.18	1.16	1.16	1.16
80	1.47	1.18	1.19	1.18	1.47

Table 5.37. Training times for the SVR-L on MP without ACK's using different number of folds

Fold no.	Training times (s)			
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>IWT</i>
10	2241	81.3	275.8	127.4
20	2053	83.3	277.4	129.4
30	2089	83.4	279.8	129.9
40	2135	83.7	279.1	129.3
50	2201	83.1	278.7	129.4
60	2044	83.2	278.5	130.0
70	2155	83.5	279.4	130.4
80	2289	83.4	278.6	130.3
				282.8
				285.4
				285.2
				284.5
				284.9
				285.4
				283.4
				285.0

Table 5.38. Training times for the SVR-RBF on MP without ACK's using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	145.8	514.7	495.4	460.3	266.7
20	145.8	514.6	495.4	460.3	266.6
30	145.8	514.6	495.4	460.4	266.5
40	145.9	514.6	495.4	460.4	266.6
50	145.8	514.7	495.4	460.4	266.5
60	145.9	514.7	495.4	460.4	266.5
70	145.9	514.6	495.4	460.3	266.6
80	145.9	514.7	495.4	460.4	266.5

Table 5.39. Training times for the MFANN on hybrid MP using different number of folds

Fold no.	Training Time(s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	2.71	2.69	2.69	2.7	2.7
20	2.73	2.75	2.47	2.55	2.74
30	2.49	2.73	2.53	2.69	2.52
40	2.74	2.72	2.73	2.51	2.73
50	3.1	2.73	2.51	2.71	2.48
60	2.72	2.47	2.69	2.68	2.5
70	2.72	2.71	2.71	2.61	2.53
80	2.74	2.52	2.72	2.73	2.5

Table 5.40. Training times for the SVR-L on hybridMP using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	1879	105.7	344.1	177.0	252.3
20	1711	108.2	345.9	179.9	254.6
30	1752	108.3	349.0	180.5	253.4
40	1789	107.8	348.2	179.7	253.9
50	1845	107.9	347.6	179.9	254.2
60	1714	108.1	347.3	180.7	254.6
70	1707	107.4	348.5	181.2	252.9
80	1819	108.3	348.5	181.1	254.3

Table 5.41. Training times for the SVR-RBF on hybrid MP using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	478.6	850.7	563.5	987.1	373.9
20	478.6	849.3	563.3	987.1	373.7
30	478.6	850.3	563.3	988.2	373.7
40	480.3	850.7	563.3	988.2	373.8
50	478.6	850.7	563.5	988.2	373.7
60	480.3	850.7	563.3	988.2	373.8
70	480.3	850.7	563.3	987.1	373.9
80	480.3	850.7	563.5	988.2	373.7

Table 5.42. Training times for the MFANN on DSM using different number of folds

Fold no.	Training Time(s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	0.13	0.01	0.01	0.01	0.01
20	0.01	0.01	0.03	0.01	0.01
30	0.01	0.02	0.001	0.02	0.01
40	0.01	0.01	0.01	0.01	0.01
50	0.02	0.01	0.01	0.01	0.01
60	0.01	0.01	0.01	0.01	0.01
70	0.01	0.01	0.01	0.02	0.01
80	0.01	0.01	0.01	0.01	0.01

Table 5.43. Training times for the SVR-L on DSM using different number of folds

Fold no.	Training times (s)			
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>IWT</i>
10	453.8	373.9	467.2	533.4
20	451.1	371.7	464.4	531.3
30	450.0	370.8	465.3	528.0
40	451.8	372.2	465.1	531.1
50	451.0	371.6	464.4	528.2
60	453.7	373.8	465.2	533.4
70	451.2	371.7	464.5	530.4
80	452.6	372.9	465.9	532.0

Table 5.44. Training times for the SVR-RBF on DSM using different number of folds

Fold no.	Training times (s)				
	<i>CWT</i>	<i>CU</i>	<i>NRT</i>	<i>PU</i>	<i>IWT</i>
10	467.4	311.6	560.2	373.9	410.3
20	464.6	309.7	556.9	371.7	407.9
30	463.5	309.0	555.6	370.8	406.9
40	465.3	310.2	557.7	372.2	408.5
50	464.5	309.7	556.8	371.6	407.8
60	467.3	311.5	560.2	373.8	410.3
70	464.7	309.8	557.0	371.7	408.0
80	466.1	310.7	558.7	372.9	409.2

6. CONCLUSION

In this thesis, a reliable methodology to predict the performance measures of a multiprocessor interconnection network using machine learning tools is proposed. This thesis proposes to use MFANN's to predict the performance measures of a MP and DSM multiprocessor architecture. The basic idea is to collect a small number of performance measures by using a statistical simulation and predict the performance of the system for a large set of input parameters based on these. The important input parameters of the simulation based on the architecture protocol type and they are: the *number of nodes*, the *number of the threads* executed by each processor, *ratio of the mean thread run time to channel transfer time*, *thread run time*, *protocol type* and *pattern* of the destination node selection changes to represent: UN, HR, BR and PS. The obtained dataset contains five output performance measures (i.e. *NRT*, *CWT*, *PU*, *CU* and *IWT*) of the architecture.

Opnet Modeler is used to statistically simulate both the MP and DSM models to produce the training and testing datasets. The obtained data set as a result of the statistical simulation consists of four different sets based on the protocol types and they are: a) message with ACK's (792 data points); b) MP and without ACK's (792 data points); c) hybrid message passing (1584 data points); and d) distributed shared memory dataset (792 data points). Using different CV for the folds numbers, the performance measures for correlation coefficients R and the error metrics for SEE , MAE , RAE and $RRSE$ have been considered. MFANN, SVR, GRNN and MLR models with different number of folds have been developed to predict these performance measures. R , SEE , MAE , RAE and $RRSE$ values of the developed models have been calculated.

Employing MP paradigm, for all performance measures, the MFANN-based prediction model performs better (i.e., higher R and lower SEE , MAE , RAE and $RRSE$) than SVR-based, GRNN-based and MLR-based prediction models, the SEE for the MFANN-based prediction model decreases as the number of folds in the test set increases from 80 down to 10. However, it is observed that the SEE of the ANN-based model increases as the number of folds exceeds 80. The prediction errors

related to *NRT* and *IWT* are higher than the ones related to *CWT*. This is because of the high standard deviation of *NRT* and *IWT* in the dataset. The *R* values for prediction of *CWT*, *CU*, *NRT*, *PU*, and *IWT* are limit to 1 for all folds. In general, prediction models for MP with no ACK's protocol perform better than the ones for both Hybrid MP and MP with ACK's. MFANN records the highest results using 80 folds CV. Hybrid MP prediction models perform better than the ones for MP with ACK's. Increasing the CV fold numbers enhances the performance of all the machine learning models.

Using DSM protocol the study outlines the following notes: Machine learning methods can be organized descendingly based on their accuracy as: SVR-L, SVR-RBF, MLR, GRNN and MFANN. Based on training and testing execution duration times, machine learning techniques can be ordered as: MFANN, SVR-RBF, SVR-L and MLR-based. Because of the non-existence of the training phase in GRNN-based, it performs the results faster than the other methods. Considering the results accomplished using the CV method across all folds from 10 to 80 folds, the 80 CV fold usually shows the best results for MFANN, SVR-RBF and the MLR model. MLR-based models relatively show similar results compared with the robust modern machine learning methods. In order to predict the *IWT*, GRNN, MFANN and SVR-RBF models are reliable to be used, whereas the linear (SVR-L and MLR) models fail compete to the robust methods. For *NRT*, the linear models (SVR-L and MLR) are not advisable to be used for predicting *NRT* for the DSM protocol.

The findings obtained by this study demonstrate the benefits of employing machine learning techniques on a multiprocessor interconnection network architecture, which can be optimized for the types of communication inherent in the domains of MP and DSM, namely estimate efficient performance criteria's of relatively large-scale system. The techniques implemented within such a framework has the potential to realize not only an increase in the level of performance improvement of the system but also a simultaneous increase in the performance of the most dominant programming models (MP and DSM).

Future work can be performed in a number of areas. The first area would be expanding the number of input parameters in the dataset. The second area would be

feature extraction on input variables. In this case, the critical attributes that best predict performance measures can be selected from a candidate set of attributes through feature selection algorithms combined with MFANN's.

REFERENCES

- ACACIO, M.E., GONZÁLEZ, J., GARCÍA, J.M. and DUATO, J., 2002. The use of prediction for accelerating upgrade misses in CC-NUMA multiprocessors. Proc 11th International Conference on Parallel Architectures and Compilation Techniques (PACT'02), 155.
- ACI, C. I. and AKAY M. F., 2010. A new congestion control algorithm for improving the performance of a broadcast-based multiprocessor architecture. Journal of Parallel and Distributed Computing, 70(9):930-940.
- AKAY, M. F. and ABASIKELEŞ I., 2010. Predicting the performance measures of an optical distributed shared memory multiprocessor by using support vector regression. Expert Systems with Applications, 37:6293-630.
- AKAY, M. F. and ZAYID ELRASHEED.I.M., 2011. Predicting the performance measures of a message passing multiprocessor architecture by using artificial neural networks. 2nd International Symposium on Computing in Science and Engineering,ISCSE-2011. June 1- 4, Kuşadası, Turkey. pp. 53-58.
- AKAY, M. F., and KATSINIS C., 2007. Performance improvement of parallel programs on a broadcast-based distributed shared memory multiprocessor by simulation. Simulation Modelling Practice and Theory, 16 (2008): 338–352.
- ALAM, S.R., BARRETT, R.F., FAHEY, M.R., KUEHN, J.A., MESSER, O.E.B., MILLS, R.T., ROTH, P.C., VETTER, J.S. and WORLEY, P.H., 2008. An Evaluation of the Oak Ridge National Laboratory Cray XT3. International Journal of High Performance Computing Applications, 22:52-80.
- ALONSOA, J.M., IZUB C. and GREGORIO J.A., 2008. Improving the performance of large interconnection networks using congestion-control mechanisms. Performance Evaluation, (2008):203-211.
- ALPAYDIN, E., 2010. Introduction to Machine Learning. 2nd Edition. MIT press. London, UK.

- BANI-MOHAMMAD, SAAD, ABABNEHA, ISMAIL and HAMDAN, MAZEN, 2011. Performance evaluation of noncontiguous allocation algorithms for 2D mesh interconnection networks, *The Journal of Systems and Software*, 84:2156– 2170.
- BLACK, B., HUANG, A.S., Lipasti, M.H., Shen, J.P., 1996. Can trace-driven simulators accurately predict superscalar performance? In: *Proc. Int'l Conf. Computer Design, ICCD*, pp. 478–485.
- BLUMRICH, M., CHEN, D., COTEUS, P., GARA, A., GIAMPAPA, M., HEIDELBERGER, P., SINGH, S., STEINMACHER-BUROW, B., TAKKEN, T., VRANAS, P., 2003. Design and analysis of the bluegene/L torus interconnection network. IBM Research Report RC23025 (W0312-022).
- BOUZID, A. and ABUSHAGUR M.A.G., 1996. Thin-film approximate modeling of in-core fiber gratings, *Opt. Eng.*, 35 (10):2793–2797.
- CAO, JIANNONG, BENNETT, GRAEME, ZHANG, KANG, 2000. Direct execution simulation of load balancing algorithms with real workload distribution, *The Journal of Systems and Software*, 54: 227-237
- CHAUDHURI, M., HEINRICH, M., HOLT, C., 2003. Latency, Occupancy, and Bandwidth in DSM Multiprocessors: A Performance Evaluation. *IEEE Transactions on Computers*, 52(7):862-880.
- CHEN, M.S., AND YEN H.W., 2011. Applications of machine learning approach on multi-queue message scheduling. *Expert Systems with Applications*, 38:3323–3335.
- CHERKASSKY, V. and MA Y., 2004. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17:113–126.
- CHOU, Y., FAHS, B., ABRAHAM, S., 2004. Microarchitecture optimizations for exploiting memory-level parallelism, in: *Proc. Int'l Symp. Computer Architecture, ISCA*, pp. 76–87.
- CHUNG, Y., KIM, H., PARK, JIN-WON and LEE, K., 2001. Performance evaluation for CC-NUMA multiprocessors using OLTP workload, *Microprocessors and Microsystems*, 25:221-229.

- COHEN, W.E., HYDE, D.W. and GAEDE R.K., 2000. An Optical Bus-Based Distributed Dynamic Barrier Mechanism, *IEEE Transactions on Computers*, 49(12):1354-1365
- CULLER, D., SINGH J. P. and GUPTA A., 1999. *Parallel Computer Architecture: A Hardware/Software Approach*. Fourth Edition Morgan Kaufmann Publishers San Francisco, USA.
- DALLY, W.J., and TOWLES, B., 2004. *Principles and Practices on Interconnection Networks*. Morgan Kaufmann, 550 p.
- DRAPER, NORMAN , R. and SMITH HARRY, 1998, *Applied Regression Analysis*. Third Edition. Wiley Copyright. London, UK.
- DUATO, J., YALAMANCHILI, S. and NI. L., 2003. *Interconnection Networks: An Engineering Approach*. International Edition. Morgan Kaufmann Publishers. USA.
- EECKHOUT, L., SAMPSON J. and CALDER B., 2005. Exploiting program microarchitecture independent characteristics and phase behavior for reduced benchmark suite simulation. In *Proceedings of the IEEE international workload characterization symposium*, pp. 2–12.
- EISLEY, N., PEH, L.S. and SHANG, L., 2006. In-Network Cache Coherence. *IEEE Computer Architecture Letters*, 5:34-37.
- EL-REWINI HESHAM and ABD-EL-BARR MOSTAFA, 2005. *Advanced Computer Architecture and Parallel Processing*. John Wiley & Sons, Inc. Publication. New Jersey, USA.(5):129- 230.
- FANG, ZHEN, ZHANG, LIXIN , CARTER, JOHN B., CHENG, LIQUN , PARKER, MICHAEL, 2005. Fast synchronization on shared-memory multiprocessors: An architectural approach *J. Parallel Distrib. Comput.* 65:1158 – 1170.
- FIRAT, M. and GUNGOR M., 2009. Generalized regression neural networks and feed forward neural networks for prediction of scour depth around bridge piers. *Advances in Engineering Software*, 40:731–737.

- GENBRUGGE, D. and EECKHOUT L., 2007. Statistical simulation of chip multiprocessors running multi-program workloads. Proc. of the 25th International Conference on Computer Design. ICCD'2007. IEEE. October, 7-10, Lake Tahoe, CA. pp. 464–471.
- GUNN, S. R., 1998. Support vector machines for classification and regression. Technical Report. Department of Electronics and Computer Science, University of Southampton, UK.
- HECHT, D. and KATSINIS C., 2003. Performance Analysis of a Fault-tolerant Distributed-shared-memory Protocol on the SOME-Bus Multiprocessor Architecture, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), United States, 213.
- HECHT, D., 2002. Fault-Tolerant Distributed Shared Memory on a Broadcast-based Interconnection Architecture. PhD dissertation. Dept of EEEng, Faculty of Drexel University, Philadelphia.pp.7-14.
- HEIRMAN, W., DAMBRE, J., VAN CAMPENHOUT, J., 2007. Synthetic Traffic Generation as a Tool for Dynamic Interconnect Evaluation. ACM Press, 65-72.
- HEMENWAY, R., 2008. High Bandwidth, Low Latency, Burst-Mode Optical Interconnect for High Performance Computing Systems, IEEE, 1(1):4.
- HENNESSY, J. and PATTERSON, D., 2007. Computer architecture: a quantitative approach. Fourth Edition. Morgan Kaufmann Publisher. San Francisco, CA. pp.196-264.
- HU, S. and JOHN, L., 2006. Avoiding store misses to fully modified cache blocks. Proc. 25th IEEE Int. Performance, Computing, and Communications Conference (IPCCC'2006):286-296.
- KATSINIS, C., 1998. Performance Analysis and Simulation of the SOME-Bus Architecture Using Message Passing. IEEE, 1998: 68-72.
- KATSINIS, C., 2001. Performance analysis of the simultaneous optical multiprocessor exchange bus. Parallel Computing, 27(8):1079–1115.
- KATSINIS, C., 2004. A Scalable Interconnection Network Architecture for Petaflops Computing. The Journal of Supercomputing, 27:103–128.

- KHASHEI M, HAMADANI A. Z. and BIJARI B., 2012. A novel hybrid classification model of artificial neural networks and multiple linear regression models. *Expert Systems with Applications*, 39:2606-2620.
- KULICK , J., COHEN, W. E., KATSINIS, C., WELLS, E., THOMSEN, A., GAEDE, R. K., LINDQUIST, R. G., NORDIN, G. P., ABUSHAGUR, M. and SHEN, D., 1995. The Simultaneous Optical Multiprocessor Exchange Bus. *IEEE Xplore*. pp. 336- 344.
- KUROSE, JAMES F. and ROSS, KEITH W., 2010. *Computer Networking: A Top-Down Approach*. Fifth Edition. Pearson Education Inc. Boston, MA 02116. pp. 111 – 463.
- LEE, H., JIN, L., LEE, K., S. DEMETRIADES, M. MOENG, S. CHO, 2010. Two-phase tracedriven simulation (TPTS): a fast multicore processor architecture simulation approach, *Software: Practice and Experience (SPE)* 40 (3):239–258.
- LEE, K., CHO, S., 2012. Accurately modeling superscalar processor performance with reduced trace, *J. Parallel Distrib. Comput.*,(2012), doi:10.1016/j.pdc.2012.12002
- LI, Y., LEE, B., BROOKS, D., HU, Z., SKADRON, K., 2006. CMP design space exploration subject to physical constraints, in: *Proc. Int'l Symp. High-Performance Computer Architecture, HPCA*, pp. 62–72.
- NUSSBAUM, S., and SMITH, J. E., 2002. Statistical simulation of symmetric multiprocessor systems. In *Proc of the 35th annual simulation symposium*. pp. 89–97.
- OPNET Inc., 2012. *OPNET Modeler*. OPNET University program, http://www.opnet.com/university_program.
- RUI, H., ZHANG, LONGBIN, HU WEIWU, 2007. Accelerating sequential programs on Chip Multiprocessors via Dynamic Prefetching Thread Microprocessors and Microsystems, 31:200–211
- SCHÖLKOPF, B. and SMOLA, A. J., 2002. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press. Cambridge, MA.

- SENDAG, R., YILMAZER, A., YI, J. J. and UHT, A. K., 2007. The impact of wrong-path memory references in cache-coherent multiprocessor systems. *Journal of Parallel and Distributed Computing*, 67:1256–1269.
- SHIN, J. and PINKSTON, T.M., 2003. The Performance of Routing Algorithms under Bursty Traffic Loads. *Proc. Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA '03)*:737-743.
- SINGH, J.P., WEBER, W., GUPTA, A., 1992. SPLASH: Stanford parallel applications for shared memory. *Computer Architecture News*, 20(1):5–44.
- SPECHT, D. F., 1991. A Generalized Regression Neural Network. *IEEE Transactions on Neural Networks*, 2(6):568-576.
- THIELE, L., WANDELER, E. and CHAKRABORTY, S., 2005. Performance analysis of multiprocessor DSPs: A stream-oriented component model. *IEEE Signal Processing Magazine*, 22:38–46.
- UHLIG, R.A., MUDGE, T.N., 1997. Trace-driven memory simulation: a survey, *ACM Computing Surveys* 29 (2):128–170.
- VAPNIK, V.N., 2000. *The nature of statistical learning theory*. Springer. New York, USA.
- WACHOWIAK, M. P., Elmaghraby, A. S., Smolikova, R. and Zurada, J. M., 2001. Generalized regression neural networks for biomedical image interpolation. *Proc. Int. Joint Conf. on Neural Networks*. Washington DC, USA. pp. 2133-2138.
- Wang, Zhonglei and Herkersdorf, Andreas, 2010. Software performance simulation strategies for high-level embedded system design, *Performance Evaluation*, 67:717-739.
- WENİSCH, T. F., WUNDERLİCH , R. E., FALSAFİ , B. and HOE, J. C., 2006. Statistical sampling of microarchitecture simulation. In *Proc of the 20th parallel and distributed processing symposium*. April, 25 – 29, Rhodes Island, Greece.pp. 327.
- WOLF, MARİLYN, 2012. *Computers as Components: Principles of Embedded Computing System Design*. Third Edition. Morgan Kaufman, New York, USA. pp.409-457.

- YI, J.J., EECKHOUT, L., LILJA, D.J., CALDER, B., JOHN, L.K., SMITH, J.E., 2006. The future of simulation: a field of dreams, IEEE Computer 39 (11):22–29.
- ZAYID, ELRASHEED I. M. and AKAY, M. FATIH, 2012a. Computing and estimating the performance measures of a message passing multiprocessor architecture by using artificial neural networks. 2nd International Conference On Computation For Science And Technology. ICCST-2. July 9-11, Niğde, Turkey. pp.76-77.
- ZAYID, ELRASHEED I. M. and AKAY, M. FATIH, 2012b. Multilayer feed forward neural network models for predicting the performance measures of a message passing architecture. 7th International Symposium on Electrical and Computer Systems. November 29-30, Gemikonagi, Cyprus.
- ZAYID, ELRASHEED I. M. and AKAY, M. FATIH, 2012c. Predicting the performance measures of a message-passing multiprocessor architecture using artificial neural networks. Neural Comput & Applic, 21(8):DOI 10.1007/s00521-012-1267-9.
- ZHU, M., KATSINIS, C., CAI, W. and LEE, B., 2004. Key messaging on SOME-Bus clusters, Parallel Computing (2004) 947-971.

BIOGRAPHY

Elrasheed Ismail Mohommoud ZAYID was born, in Adyla Province in Darfur State western Sudan in 1972.

He received his B.Sc. degree with honors in Computer Science from Alneelain University, Khartoum, Sudan in 1998. He joined the Department of Computer Engineering of the University of Elimam Elmahdi as a teaching assistant in 1999.

He received his M.Sc. degree at the Department of Electrical and Electronics Engineering of the University of Khartoum, Sudan in March 2003. Since March 2003, he has been a lecturer at the Department of Computer Engineering of the University of Elimam Elmahdi. While pursuing his graduate studies, he held a teaching and research assistantship and has extensive teaching experience in the areas of networks architecture and computer system. In 2004 he was designated as a director for the Computer Center and was a leader of the team that established the University network system.

In December 2007, he received a Ph.D scholarship offered by both the Turkish Government and the Ministry of Higher Education and Scientific Research Sudan. In order to learn Turkish language he joined Ankara University Language Center “TÖMER” from January until July 2008. In October 2008, he was registered as a Ph.D student in the Department of Electrical and Electronics Engineering of Cukurova University.

He has co-authored two journal and four International conference papers. He is currently a Ph.D. candidate in the Department of Electrical and Electronics Engineering of Cukurova University. His research interests are computer networks and multiprocessor architectures.

Elrasheed is married and a father of two children, his son Anas and his toddler daughter Aya.