1993

Performance evaluation of an auction-based manufacturing cell using Generalized Stochastic Petri Nets.

Qiong. Zhou
University of Windsor

PERFORMANCE EVALUATION OF AN AUCTION-BASED MANUFACTURING CELL USING GENERALIZED STOCHASTIC PETRI NETS

by

Qiong Zhou

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the Department of Industrial Engineering
in Partial Fulfilment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

1993

ISBN 0-315-87344-2

Name [illegible]

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

INDUSTRIAL ENGINEERING | 0546 | U·M·I

SUBJECT TERM | SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

ABSTRACT OF THE THESIS

PERFORMANCE EVALUATION OF AN AUCTION BASED MANUFACTURING CELL USING GENERALIZED STOCHASTIC PETRI NETS

by

Qiong Zhou

Master of Applied Science in Industrial Engineering, 1993

University of Windsor, Windsor, Ontario, Canada, N9B 3P4.

Advisor: Dr. Sourin P. Dutta

An auction-based manufacturing system is characterized by the fact that the central control computer is eliminated and all the system entities make decisions locally through the negotiation process. The performance evaluation of such manufacturing systems is a new research topic because the traditional OR scheduling and dispatching models usually assume the existence of a central controller, which is, however, removed from an auction-based system. This thesis deals with the modelling and analysis of an auction-based flexible manufacturing cell, in which multiple machining centres are organized in two stages and are capable of processing multiple part types. The Generalized Stochastic Petri Nets is used as the modelling tool to built general models for the FMC with/without instage buffers. Performance measures such as, machine throughput and utilizations, number of block jobs, and queue lengths in the buffers are obtained by solving the model using SPNP package. The operational behaviour of the heterarchically controlled cell is discussed, *i.e.*, the auction behaviour of downstream machines, the influence of instage buffers, and the effect of the number of part types.

To my husband, Chuan-Zhang Tang, and my son, Jia-Hua Tang

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Dr. S.P. Dutta, for his valuable guidance throughout the entire progress of my thesis. I am most grateful for the help provided by Dr. M.H. Wang, who allowed me to have frequent discussions with him. Special thanks to Dr. M. Ahmadi of Electrical Engineering for his interesting comments and opinions on all the relevant topics of this thesis. I am indebted to Dr. R.S. Lashkari for his suggestions that substantially improved both the presentation and quality of this research. A special note of thanks to Ms. Jacquie Mummery and Mr. Tom Williams for their continuous assistance while my conducting this research.

My sincere thanks to Dr. D.M. Upton of Harvard University for providing me useful information and documents on heterarchical control architectures. Much appreciation also goes to Dr. K.S. Trivedi and Mr. Hoon Choi of Duke University for the installation of the SPNP software package and their immediate responses to my questions.

I am grateful to the National Sciences and Engineering Research Council of Canada for support of this work under grant number OGP00004141.

Finally, I wish to express my sincere appreciation to my husband for his constant encouragement and understanding and to my parents for their care and affection. Without this encouragement this work would never have been completed. During this research many events took place in my life. The one that I have grown to enjoy and appreciate the most was the growth of my son, Jiahua.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

BT(i,j): bid time of downstream machine i for part type j

E_j: enabling function of transition j

F(i): the set of machines with free buffer places

H: the set of enabled transitions in marking M

K_i: size of buffer i

m: number of upstream machines

M: marking

n: number of downstream machines

P: place

$|P|$: number of tokens in place P

RS(t): random switch function for transition t

t: transition

TH(t): average throughput of transition t

TH1(j): average throughput of upstream machine j

TH2(i, j): average throughput of downstream machine i for part type j

TH2(i): average throughput of downstream machine i for all part types

UT1(j): utilization of upstream machine j

UT2(i): utilization of downstream machine i

μ_{ij}: processing rate of machine i for part type j

λ: arrival intensity

Abbreviations

AGV: automated guided vehicle

buffer i: instage buffer of downstream machine i

CSPN: coloured stochastic Petri nets

CTMC: continuous time Markov chain

DSPN: Petri nets with deterministic and stochastic times

ESPN: extended stochastic Petri nets

FIFO: first in first out

FMC: flexible manufacturing cell

FMS: flexible manufacturing system

GSPN: generalized stochastic Petri nets

MC: machining centre

MC j: upstream machine j

MC i: downstream machine i

OR: operational research

PN: Petri nets

SPN: stochastic Petri nets

SPNP: stochastic Petri net package

TPN: timed Petri nets

TPPN: timed place Petri nets

TTPN: timed transition Petri nets

Chapter 1

INTRODUCTION

1.1 Overview

An *automated manufacturing system* (AMS) is an interconnected system of material handling and processing stations capable of automatically processing a wide variety of part types simultaneously under computer control. The *control system*, whose functionality is established by the *control architecture*, tries to optimize the AMS performance while achieving production requirements. Usually, the operational control of an AMS is very complicated. With increasing system size, the complexity of a *hierarchically* controlled AMS tends to grow rapidly, which results in the deterioration of its designability, maintainability, expendability, and fault tolerance. As an alternative, *heterarchical* control architectures (also known as auction-based control strategies) offer prospects of reduced complexity, high flexibility, and improved fault tolerance.

Some research on heterarchical control architectures has been carried out during the past ten years, including: feasible negotiation procedures, experimental implementations, and performance analysis. Performance evaluation of such control strategies is a relatively new research topic because the traditional OR (operational research) scheduling and dispatching models usually assume the existence of a central controller with a global database, which is however, eliminated in an auction-based manufacturing system.

Numerical results reported so far on the performance of heterarchically controlled systems have been based on simulation models for systems consisting of machines, each dedicated to a single operation. No results have been published on such a system with *multiple part types and multiple machining centres*. As a matter of fact, in an automated manufacturing system, a machining centre is capable of performing more than one operation and usually multiple part types exist in the system simultaneously. Consequently, the control problem for these types of systems becomes much more complicated. The work of this thesis can be regarded as one of the initial efforts towards the solutions to this problem.

Analytical methods and simulation models are traditional approaches for performance analysis of manufacturing systems. Recently, Petri nets have been found to be a useful tool for the modelling and analysis of discrete-event dynamic systems. They are particularly valuable when state and control information are distributed throughout the system. Due to the nature of heterarchical control strategies, concurrency and synchronization, Petri nets, specifically, Generalized Stochastic Petri Nets, are used as a modelling tool for this work.

1.2 Objectives

The objective of this thesis is to develop Petri net models of an auction-based Flexible Manufacturing Cell (FMC) with *multiple part types and multiple machining centres.* Based on the models developed, the auction behaviour of the FMC will be investigated.

1.3 Thesis Organization

This thesis consists of 6 chapters. Chapter 2 begins by comparing the pros and cons of hierarchial control and heterarchical control to show the motivations for investigating auction-based manufacturing systems. A survey on heterarchical control is also provided.

Chapter 3 is concerned with fundamentals of Petri nets, the modelling tool applied in this work. Some questions are discussed, such as why Petri nets are used, what Petri nets are, and how Petri nets work, with focus on the Generalized Stochastic Petri Nets (GSPN).

With the background of heterarchical control and Petri nets, Chapter 4 focuses on the development of Petri net models of an auction-based FMC.

Two examples are given in chapter 5, showing how performance analysis can be carried out by the developed Petri net models. The operational behaviour of the FMC is also discussed.

Chapter 6 concludes the thesis by stating the results and outlining the future research.

Chapter 2

HETERARCHICAL CONTROL ARCHITECTURES

2.1 Introduction

The *control system* of an AMS coordinates and directs the part handling and processing activities that transform raw materials into finished products. As a result, the control system embodies many decision making responsibilities including part scheduling, part routing, and resource allocations within the manufacturing facility.

A *control architecture* creates a control system from control components. It allocates the decision making responsibilities to specific control components and determines the interrelationships between the control components. For example, a control architecture may specify one control component with responsibility for part scheduling, and another for the movement of parts. Also, the interaction between the two components can be stipulated.

Thus, the ability of the control system to carry out effective decision making will be a function of how those decision making responsibilities are divided and coordinated. In other words, the control architectures determine the effectiveness of control systems.

The earliest control architecture for manufacturing systems employed a *centralized* approach. Driven by the demands of AMSs for increased reliability/fault-tolerance, modifiability/extensibility, and reconfigurability/adaptability, and with the development of computer technology and advanced manufacturing facilities, control architectures have evolved from a traditional centralized form, to a *hierarchical* form, and recently to a *heterarchical* form. **Figure 2.1** shows the structures of the three basic forms of control architectures.

The centralized control architecture for an entire manufacturing system is no longer common. The most widely applied control strategy nowadays is the hierarchical control architecture. However, due to its increasing complexity with system size, the hierarchical control method has been challenged recently by the heterarchical control approach. In this chapter, a comparison between hierarchical control and heterarchical control is presented to show the motivations of this research. The similarity of computer organization and the control architectures for manufacturing will be discussed. Finally, a literature review on heterarchical control is presented.

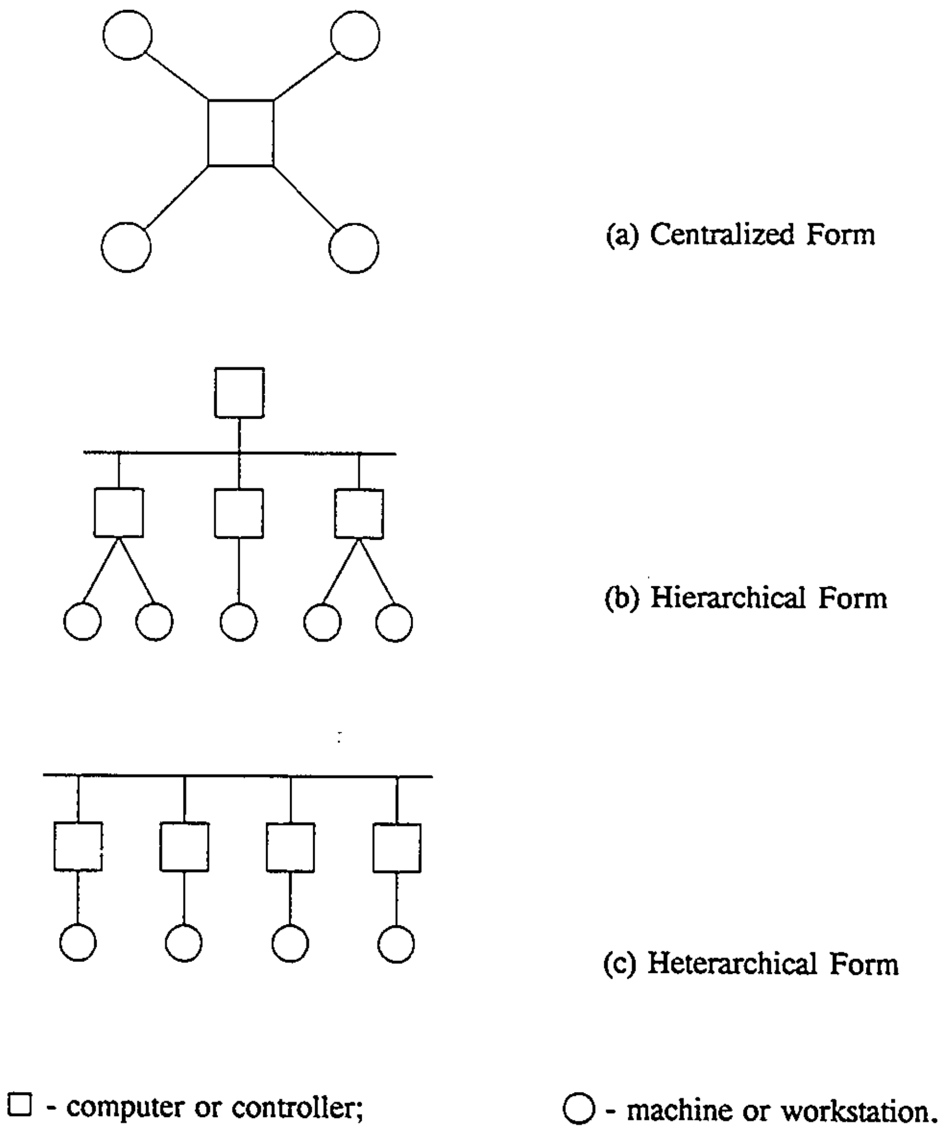


Figure 2.1 Basic Forms of Control Architectures for Manufacturing Systems

2.2 Heterarchical Control vs. Hierarchical Control

The comparison will be made according to the system modifiability, reconfigurability, and faut-tolerance, which are some of the requirements of AMS that must be met by control architectures.

A system is said to be modifiable if changes to existing elements of the system may be easily made (Cutosky *et al.* 1984).

The dynamic runtime reconfiguration of an AMS is to accommodate the auction or removal of various manufacturing systems components while the system is operational (Dilts *et al.* 1990).

Fault-tolerance indicates the ability of a system to continue to function, perhaps in a degraded state, despite the occurrence of system failures (Booth 1981).

2.2.1 Hierarchical Control

The hierarchical control architecture is characterized by "a philosophy of 'levels' of control and contains a number of control modules arranged in a pyramidal structure." (Duffie *et al.* 1988). Rigid master/slave relationships exist between decision making levels with the control decisions operated top-down and status reported bottom-up.

Hierarchical control structures are the most commonly used nowadays in manufacturing systems. However, they were challenged recently by the heterarchical control strategy because of the following reasons.

1) Software Complexity Flexible Manufacturing Systems (FMS) are typical systems of hierarchical control. Although they have brought tremendous benefits in reduced inventory, improved quality, and shortened lead time, the complexity of the control software has prevented them from being widely implemented. In FMS, flexibility can take a number of forms, including volume flexibility, routing flexibility, product flexibility, and others. To accommodate the flexibility, the FMS control software would be very complex and expensive, though global optimization could be achieved. The generic controller proposed by Tirpak *et al.* (1992) gives some insight into the software complexity of a hierarchically controlled Flexible Manufacturing System.

2) Modifiability and Reconfigurability The hierarchical control structure tends to be rigid and *fixed* in the early design stages for a range of *known products* under given configurations. If, *in the long term, new families of products* are introduced into the system and *new machines* are added (or an old one updated), the manufacturing system must be down in order to implement the software changes, and the modification would be very costly to make because of the complexity of the central control software. Also, it is difficult, too, to accommodate the addition or removal of various manufacturing system components while the system is operating.

3) Fault-Tolerance Because of the master/slave relationship between control levels, the failure of a master controller at a higher lever will cause the failure of all slave controllers at the lower levels. The higher the failed master controller, the greater the number of paralysed slave controllers.

4) System Size If a manufacturing system consists of a large number of computer numerically controlled (CNC) machines, it would become computationally intractable because of the complexity of the control software. In other words, hierarchical control systems are limited in size.

2.2.2 Heterarchical Control

To overcome the shortcomings associated with the traditional centralized and hierarchical control structures, a new strategy called "heterarchical control" has been proposed by several researchers. Although there are some differences among the proposed methods, the basic ideas are the same. That is, *the central control computer is eliminated and all the system entities are fully autonomous, making decisions locally through negotiation processes.*

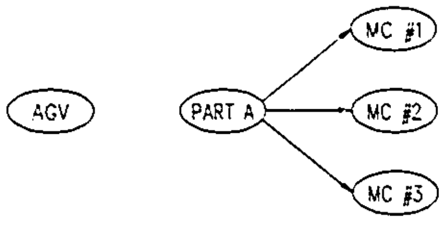
It should be noted that in traditional manufacturing systems, a part is "passive" in the sense that it cannot make any decisions about its scheduling and routing. Instead, it is processed by machines according to a schedule established by a global controller. In heterarchical manufacturing systems, a part is "active" (or "intelligent") as it has the ability of communicating with all the machines in the system by an on board radio and making decisions by an on board computer. The part's process plan has been compiled and loaded into the memory of the part's computer before the part enters the system.

With the "intelligence", a part tries to find an appropriate machine for its next operation by itself through negotiation process.

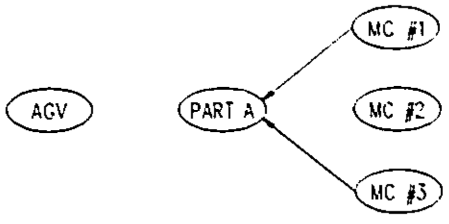
In general, the negotiation processes are divided into four phases:

1) part posting bid request
2) machine evaluation and bid generation
3) part acceptance and commitment
4) machine confirmation

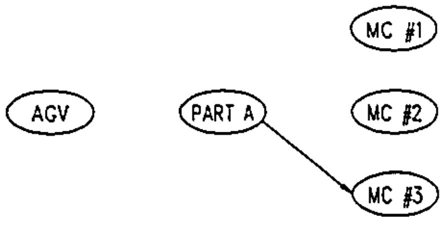
As an example, **Figure 2.2** shows a system with one part, one automated guided vehicle (AGV), and three machining centres (MCs). Part A broadcasts a request for an operation to all the machining centres in the system. One possible case would be that none of the machines is free. Then, part A has to wait a certain period of time (back-off time) and calls again. Another possibility would be that two of the machining centres are free and capable of doing the job, therefore, both respond. Part A selects the best bid, say, that of MC#3, according to certain criterion (e.g., the shortest throughput time), and makes the reservation. MC#3 confirms the reservation by sending an acknowledgement. Then part A arranges an AGV for its transportation. In a real-world situation, there could be multiple parts in a system undergoing the negotiating procedures simultaneously.



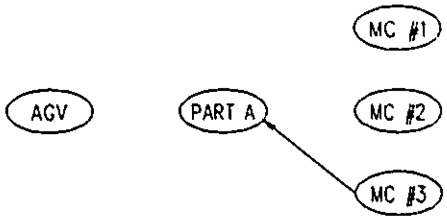
(a) Part A broadcasts a request.



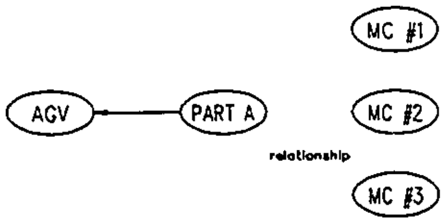
(b) Two machines are free and respond.



(c) Part A sends reservation to MC #3.



(d) MC #3 acknowledges Part A's reservation.



(e) Part A arranges transportation.

Figure 2.2 Negotiation Process of Heterarchical Control Architecture

Heterarchical architectures can improve the performance of control systems in the following aspects (Upton 1992):

1) Software Complexity From the example in **Figure 2.2**, it can be seen that a part has been processed without a central control computer, but with simple, modular, physically decentralized hardware and software.

2) Modifiability and Reconfigurability When a new machine is added into the system, there is no need to take down the system. The new machine may simply be told the "rules of the game" and becomes a part of the system with no lost production. Removing a machine from the system is also straightforward. It simply stops bidding, and jobs stop coming to it.

3) Fault-Tolerance If a machine is down, it stops responding to requests. So failures are limited to the locale where they occur such that system-wide consequences are avoided. In other words, the fault-tolerance is implicit in the design.

4) System Size Due to the elimination of the central controller and the simplicity of the distributed control strategy, the system size will no longer be limited by the complexity of control units.

Meanwhile, the disadvantages of heterarchical control are mainly two fold:

1) Local Optimization From the myopic bid structure, it can be seen that local decisions made by entities are, of course, not globally optimal. For example, a part may opt for a machine that is the most appropriate in the short term, but may find itself away from its optimal total processing path from a long term point of view.

2) Enabling Technologies The essential technologies for this type of production systems to work are chips on parts, inexpensive radios and computers on board of pallets, radio communication networks with high channel capacities, and automatic process planning, which are currently under investigations.

Although the heterarchical control architecture is somewhat restricted by the limitation of existing hardware and software technology, current research has shown that there is a growing acceptance of the concept by academics and industry leaders. **Table 2.1** is a summary of comparison between hierarchical and heterarchical control.

Table 2.1 Summary of Comparison Between Hierarchy and Heterarchy

Characteristics	Hierarchical Control	Heterarchical Control
Control architectures	rigid master/slave relationships between decision making levels	• no master/slave relationships; • full local autonomy
Software complexity	very complicated	reduced
Modifiability	difficult	easy
Reconfigurability	difficult	easy
Fault-tolerance	The higher the failure nodes, the greater the number of lower paralysed controllers.	• Local failure has no system wide influence. • implicit in the design
Optimality	global	local
Enabling technology	implemented already	under development

2.2.3 SIMD & MIMD Architectures

SIMD machine (single instruction stream - multiple data stream) and MIMD machine (multiple instruction stream - multiple data stream) are computer architectures for multiprocessor systems. One of the important features is that the entire system must be controlled by *a single integrated operating system* providing interactions between processors and their programs at various levels (Hwang, 1984). The diagrams of SIMD and MIMD are displayed in **Figure 2.3.**

SIMD: As illustrated in **Figure 2.3a**, there are multiple processing elements (PEs) supervised by the same control unit (CU). The function of the CU is to decode the instruction, segment a job into its components, and determine by which PEs these sub-jobs should be done. This is similar to the idea of *centralized* control architectures in manufacturing systems, where a central controller dispatches jobs to multiple machines.

MIMD: Unlike SIMD with only one CU for all the PUs, there is one CU in front of each PU in MIMD, as shown in **Figure 2.3b**. However, what is similar to SIMD is that all the instructions to CUs are from the central control operating system. Thus, this computer organization is similar to the *hierarchical* control structure in manufacturing systems, but different from the *heterarchical* control architectures in manufacturing.

In general, the basic ideas of SIMD and MIMD are quite similar to that of centralized control and hierarchical control for manufacturing. The schema of SIMD and MIMD could be adopted to manufacturing systems if the system configurations are the same.

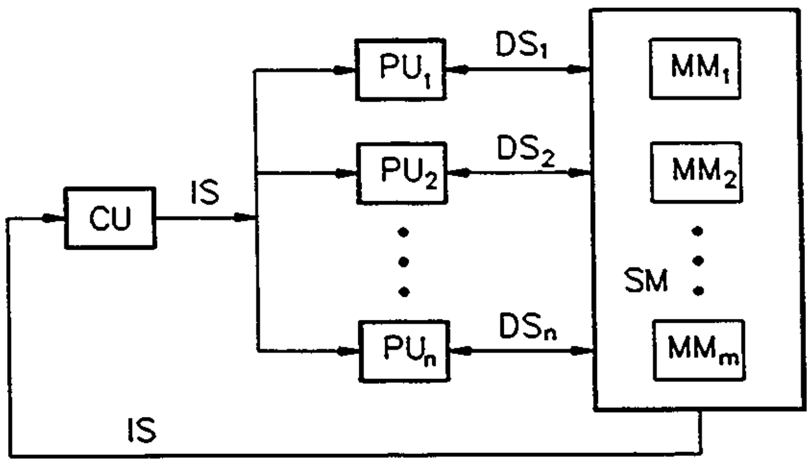


Figure 2.3a SIMD Computer

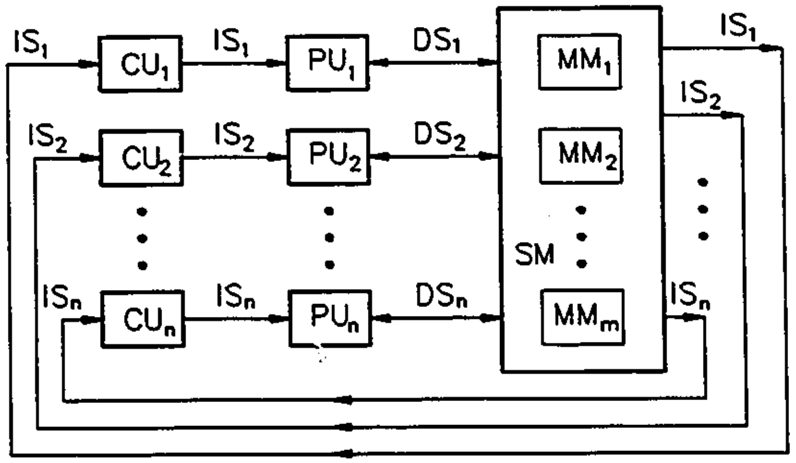


Figure 2.3b MIMD Computer

CU:	control unit	SM:	shared memory
PU:	processor unit	IS:	instruction stream
MM:	memory module	DS:	data stream

Figure 2.3 SIMD and MIMD Computer Architectures [adopted from (Hwang, 1984)]

2.3 Literature Review on Heterarchical Control

2.3.1 Feasible Negotiation Procedures

The need for an entirely new approach to the design of systems, which can be changed, adapted, expanded or updated as the situation requires, was recognized as early as in 1978 by Hatvany. A Cooperative heterarchy was suggested by Hatvany later in 1985.

Smith (1980) developed a contract negotiation scheme for cooperative problem solving, providing a starting point for the procedures in the "heterarchical" control architecture.

Since then, work has been carried out to provide feasible manufacturing operations based on the negotiation models in a variety of circumstances.

Lewis *et al.* (1982) developed a data flow of computerized manufacturing systems. Their pioneering work attempted to overcome centralized scheduling complexity in flexible manufacturing (Maley 1988).

Shaw and Whinston (1985) described a negotiation protocol used to ensure orderly information transformation and events sequencing between asynchronous, cooperating manufacturing cells. The advantages of the distributed control were explored by Shaw in 1988.

Fukuda *et al.* (1986) briefly described a hierarchical, yet distributed, control approach to overcome dynamic variations in manufacturing. They introduced a common message board to permit information exchange between control modules.

Maley (1988) proposed a system called Computer Automated Distributed Environment for Network Coordination and Execution (CADENCE), in which both the physical flow dependencies and the information flow were provided by utilizing a negotiation algorithm. CADENCE went beyond previously proposed task bidding structures by utilizing distributed decision making in managing the flow of individual intelligent parts.

Lin and Solberg (1992) proposed a generic framework for controlling the work flow in computer controlled manufacturing systems. Based on a market-like model and a combination of objective and price mechanisms, the framework allows jobs and resources to have their own intelligent controls to match individual needs.

2.3.2 Experimental Implementations

Pioneering manufacturing work was carried out by Parunak and Duffie. Parunak *et al.* (1985) utilized the actor model to meet the complexity issues of manufacturing. Their distributed system worked in a hierarchical framework but they reported that initial studies in lateral negotiation showed promise. A system, called Yet Another Manufacturing System (YAMS), has been designed to implement Smith and Davis' contract net in a manufacturing environment (Parunak 1987).

A heterarchically controlled machining cell has been constructed at the University of Wisconsin-Madison (Duffie and Piper 1986). Initial results indicated that the heterarchical approach had attractive attributes for control of flexible manufacturing systems and cells in lower development costs and improved modifiability.

Three flexible machining cell controllers have been implemented (Duffie and Piper 1987) in an effort to analyze the relative advantages and disadvantages of hierarchical and heterarchical cell control architectures. Results showed that the heterarchical approach possessed a number of advantages including increased fault-tolerance, inherent adaptability and reconfigurability, decreased complexity, and reduced software development cost.

An experimental heterarchically controlled manufacturing system has been developed, which consists of a robotic machining cell and a robotic assembly cell (Duffie, Chitturi, and Mou 1988). A heterarchical control architecture and a set of underlying design principles for developing and implementing such a system were emphasized. The design objective and philosophies for heterarchical systems were discussed further by Duffie in 1990.

2.3.3 Performance Analysis

Although various schemes have been proposed and experiments were carried out, there are few *numerical* results about the heterarchical control strategies.

M.J. Shaw (1988) describes a distributed scheduling method for cellular manufacturing systems. Simulation results showed that the bidding scheme performs better than its centralized counterpart due to the fact that by local decision making, the amount of communication activities for updating databases are reduced significantly.

Some preliminary conclusions on heterarchical control architectures have been drawn by Upton *et al.* in 1991 and 1992. The difficulties in building queuing models for auction-based manufacturing were discussed (Upton 1991). The results reported in 1991 are based on a simulation model for 1 part type and 6 machines for the same operation. The system discussed in 1992 consists of various machines, but each machine is dedicated to only one specific operation. The performance of a heterarchically controlled manufacturing system with *multiple part types and multiple machining centres* has not been addressed.

This thesis discusses the behaviour of an auction-based flexible manufacturing cell, where multiple machining centres are organized in two stages, and a part from the upstream stage will select one of the downstream machines according to the Earliest Finishing Time rule. Details of the problem will be given in chapter 4.

Chapter 3

PETRI NETS: BACKGROUND AND DEFINITIONS

3.1 Introduction

The traditional approaches for analysis of Automated Manufacturing Systems fall into two categories: analytical methods (including queuing networks and mathematical programming models) and simulation models. Recently, Petri nets, especially the Generalized Stochastic Petri Nets, have been found to be a very promising tool for performance analysis. Of course, each approach has its own advantages for system modelling. In this chapter a comparison between analytical, simulation and the Petri net models is made to show why Petri nets are chosen as the modelling tool for this research. Then, the relevant knowledge on Petri nets, especially the Generalized Stochastic Petri Nets is reviewed. Finally, the Stochastic Petri Net Package (SPNP) applied in this work is introduced.

3.2 Petri Nets as a Modelling Tool

Analytical methods use standard solution techniques and provide optimal solutions. However, restrictive assumptions are usually made in order to achieve analytical tractability. For example, by assuming that the customer interarrival time and server service time are exponentially distributed and independent of system state, the Product Form Queuing Networks are computationally very easy to solve. Nevertheless, the modelling power of these Queuing Networks is low: priority, blocking and finite buffer size are difficult to model; complex layouts, synchronization and control policies for the dispatching of customers typically cannot be handled. In addition, it is very difficult to get an intuitive understanding from the formulae which sometimes run over several pages. As a result, the method is usually used at the preliminary design stage to distinguish between alternative designs on an aggregate basis.

Discrete event simulation enables detailed description and analysis of system behaviour and is useful at all stages of design and operational analysis. However, the time required for developing a computer program will be significant if the model is detailed. The performance estimates can be accurate only if the number of simulation runs is made large. Consequently, the simulation tool turns out to be computationally expensive (Viswanadham *et al.* 1992). Moreover, although a complex system can be modelled by simulation at more detailed levels, the complex interactions cannot be understood visually from simulation.

Petri nets are modelling tools that lie between analytical and simulation methods, providing analytical results with much of the modelling flexibility of simulation. Petri nets are representatively powerful because they allow model description in a conceptually simple and graphical manner; they can exactly model non-product form features, such as priority, synchronization, blocking, splitting of customers, *etc.* The modelling power is also enhanced by their capabilities in analyzing performance quantitatively as well as in verifying models qualitatively, such as liveness, boundedness, conservativeness, and reversibility, based on reachability trees. They are computationally efficient in the sense that the mathematical foundation of Stochastic Petri Nets is Continuous Time Markov

Chain (CTMC), but the construction of the entire state space is avoided since this process has been automated by some software packages based on the reachability tree of a Petri net structure.

The limitations lie in that the net structure and the net reachability tree could be very complex in a large system. Nevertheless, it can serve as a ready simulation model if it is intractable.

In conclusion, Petri net modelling is most suitable when the researchers try to understand a system with synchronization and cooperation among concurrent processes. **Table 3.1** is a summary of comparison between queuing, simulation and Petri net models.

As will be discussed in section 4.1, the manufacturing cell under investigation has the following characteristics:

- Concurrency or parallelism: Many operations take place simultaneously.
- Conflict: More than two processes may require a common resource, such as a buffer or a machine, at the same time.
- Synchronization: The starting of machining operations must be controlled to ensure correct operation of the overall system.
- Splitting customers: Dynamic decision making is involved for dispatching of parts. Since a part chooses a machine according to the criterion of least expected delay, the dispatching of a part to a particular machine depends not only the number in its queue, but also the number in other queues, which will result in non-product form solutions.

- Non-identical customers: A machining centre has different mean processing rates for different part types.
- Finite buffer sizes: Only limited number of parts can be held in a buffer.
- Blocking: Because of finite buffer sizes and limited capacity of machining centres, a part might be blocked when its first operation is finished.

It is very difficult to built a queuing model with those properties. Simulation can be used, but Petri nets are preferred in order to comprehend the task allocation policy conceptually and theoretically.

Table 3.1 Comparison between Queuing, Simulation and Petri Net Models

	Queuing Models	PN Models	Simulation Models
Assumptions	exponential I.I.D.	general distribution	any distributions
Modelling powers	lower	higher: much of simulation's flexibility	highest
Intuitive understanding	difficult	easy	difficult
Qualitative analysis	No	Yes	No
Quantitative analysis	easy	easy with problems of reasonable sizes	expensive
Application scenarios	at the initial design stage when resource allocation mechanism is not considered	• at all design stage • most efficient for issues about synchronization, complex resource allocation, *etc.*	at all design stage

3.3 Definitions of Petri Nets

There are variations among the available definitions for Petri nets. However, the differences are notational. The format given by Ajmone *et al.* (1984) is followed in this research.

3.3.1 Standard Petri Nets

Definition 1: A *marked Petri net* is a five-tuple

$$PN=(P, T, A_i, A_o, M_0)$$

where

$P=\{p_1,p_2,\ldots,p_n\}$ is a set of *places*,

$T=\{t_1,t_2,\ldots,t_m\}$ is a set of *transitions*,

$A_i\subset(P\times T)$ is the set of *input arcs* that defines directed arcs from places to transitions,

$A_o\subset(T\times P)$ is the set of *output arcs* that defines directed arcs from transitions to places,

$A=(A_i \cup A_o)$ is the set of transition arcs, and

$M_0=\{m_{01},m_{o2},\ldots,m_{on}\}$ is the *initial marking.*

In the graphical representation of a Petri net (PN), places are drawn by circles and transitions by bars. The input and output functions are represented by directed arcs from places to transitions and vice-versa, respectively, if arcs exist between them. Tokens denoted by black dots or numbers residing in the places.

In a manufacturing system, a place is usually a shared resource or a condition for an event to occur. A transition is generally used to represent the initiation or termination of an event. Input functions and output functions establish unidirectional relationships between places and transitions, and transitions and places, respectively. Tokens in places indicate that resources are available or conditions are true.

Definition 2: A marked Petri net *executes* according to the following rules:

1. A transition is *enabled* when all its input places contain at least one token.
2. An enabled transition can *fire*, thus removing one token from each input place and placing one token in each output place.
3. Each firing of a transition modifies the distribution of tokens on places, say marking M, and thus produces a new marking, say M', for the PN. M' is called *immediately reachable* from M.

Definition 3: The *reachability set* of a Petri net $R(M_0)$ is defined as the set of all markings that are reachable from initial marking M_0.

As Standard Petri nets do not include a time concept, they are used only for qualitative and logical analysis of systems. The emergence of *Timed Petri Nets* (TPN) made it possible to perform quantitative analysis of systems. By associating a constant time delay to either places or transitions, we have *Timed Place Petri Nets* (TPPN) and *Timed Transition Petri Nets* (TTPT).

Stochastic Petri Nets (SPN) were proposed because probabilistic performance models allow the capture of the essence of the system behaviour through probabilistic assumptions so that a detailed deterministic description of the system operations can be avoided. The use of exponential distributions for the firing rates of timed transitions is particularly attractive for two reasons. First, exponential timed Petri nets can be mapped onto Continuous Time Markov Chains (CTMC), which lays the mathematical foundation of the solution algorithm for SPN (Molloy 1982). Second, the memoryless property of the exponential distribution makes it unnecessary to distinguish between the distribution of the delay itself and that of the remaining delay after a state change.

Recognizing that both time-consuming activities and logical behaviours exist in a system, *Generalized Stochastic Petri Nets* (GSPN) allow transitions to be either timed or immediate. Also, the state space generated by GSPN is smaller than that by SPN for a topologically identical PN model.

Some other extended Petri nets are: *Coloured Petri Nets*, which give more compact graphical representation of Petri Nets; *Extended Stochastic Petri Nets*, which is an effort to include non-exponential distribution in the analysis of SPN-based models; and DSPN, which are *Petri Nets with Deterministic and Exponential Firing Times*.

GSPN is chosen as the modelling tool for this thesis because it is the most extensively used class of SPN and are suitable for the problems in question. The Stochastic Petri Net Package (SPNP) (Ciardo *et al.* 1992) is available to analyze GSPN models automatically.

3.3.2 Generalized Stochastic Petri Nets

Definition 4: A *generalized stochastic Petri net* is a five-tuple

$$GSPN=(P, T, A, M_0, L)$$

where

(P, T, A, M_0) is a standard Petri Net, and $L=\{l_1, l_2, \ldots, l_{m'})$ is the set of exponential firing rates associated with m' timed transitions.

Definition 5: *Random switch* is defined as a set of immediate transitions with relative probability of firing (*switching distributions*) for resolving conflict when morc than one immediate transition is enabled at the same time.

The identification of random switches is a crucial aspect of the definition of a GSPN. Sometimes, ingenuity and insight into the system operations may be required for the definition of "correct" switching distributions in all markings.

Definition 6: The *firing rules* of GSPN are as follows:

1. If the set of enabled transition H comprises only timed transitions, then transition t_i, $i \in H$, fires with probability $$\frac{l_i}{\sum_{k \in H} l_k}$$
2. If H comprises both timed and immediate transitions, immediate ones have higher priority. If there is only one immediate transition, then this is the one that fires. When H comprises several immediate transitions, they fire according to the switching distribution.

Definition 7: *Tangible markings* are markings that enables no immediate transitions; otherwise, they are called *vanishing markings*.

The mathematical foundation of performance analysis by Stochastic Petri nets is based on the theorem due to Molloy (1982): Any finite place, finite transition, marked exponential timed (stochastic) Petri net is isomorphic to a continuous time Markov chain.

A Generalized Stochastic Petri net can still be mapped into a Markov Chain by removing vanishing markings, since they do not contribute to the measurable behaviour of the model. Therefore, the performance analysis by a Petri net model can be summarized by **Figure 3.1**. Specifically, all one has to do is to model the system with a Petri net. Then, based on the initial marking, the reachability tree can be obtained and analyzed quantitatively.

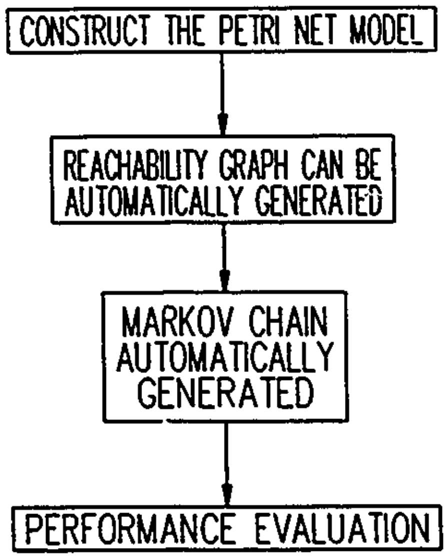

CONSTRUCT THE PETRI NET MODEL
REACHABILITY GRAPH CAN BE AUTOMATICALLY GENERATED
MARKOV CHAIN AUTOMATICALLY GENERATED
PERFORMANCE EVALUATION

Figure 3.1 Procedures of Performance Analysis by GSPN

3.3.3 Some Extensions to Petri Nets

Many extensions to the standard Petri nets were introduced to increase the modelling power of the tool. Although not all the extensions are used in this research, a list is provided for completeness. The definitions given by Ciardo (1987) follow:

Definition 8: *Multiple arcs* are arcs associated with multiplicity k, an integer number. An input arc from p to t with multiplicity k requires k tokens in p to enable t, and it causes their removal upon firing. An output arc with multiplicity k from t to p causes the addition of k tokens in p when fires.

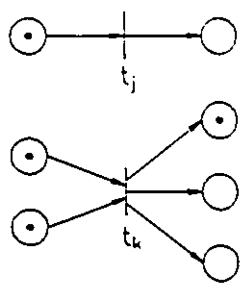
Definition 9: An *inhibitor arc* with multiplicity k from p to k disables t if k or more tokens are in p. In particular, if the multiplicity is 1, t is disabled unless p is empty.

Definition 10: If each *transition* is assigned a fixed *priority* (a non-negative integer), the enabling rule is modified so that, in each marking, only the enabled transitions with the highest priority are really enabled, while the remaining ones are disabled.

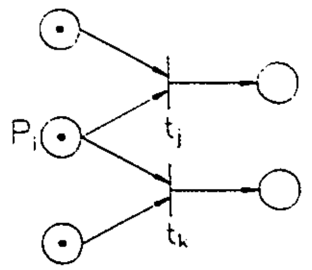
Definition 11: An *enabling function* E_t can be defined on each transition t. If $E_t(m)=1$, t is enabled in marking M, otherwise t is disabled.

Definition 12: Enabling functions, firing rates, switching distributions, and arc multiplicity can be different in each marking. This is called *marking dependent*.

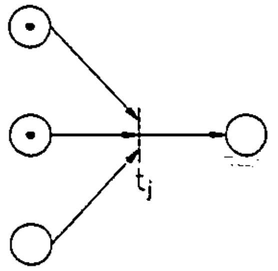
All these additional structures or the combination of these structures are used to selectively disable a transition in a marking which would otherwise enable it. The following examples, **Figure 3.2a-f,** show some of the modelling power of petri net models with the aid of the above-mentioned extensions.



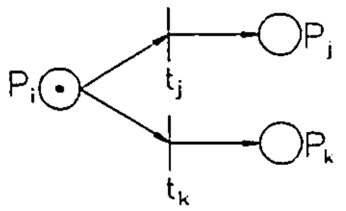
a) **Concurrency** The two transitions t_j and t_k can fire in any order.



b) **Conflict** Transitions t_j and t_k are in conflict since firing either will remover the token from P_i, disabling the other transition.

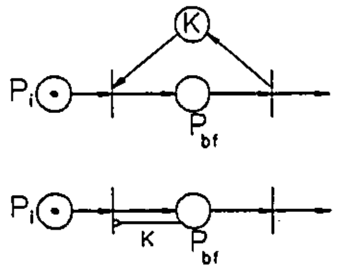


c) **Synchronization** Transition t_j will not be enabled until a token arrives at the place currently without a token.

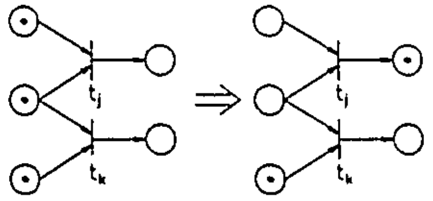


d) **Splitting customers** A token in P_i is dispatched to either process t_j or t_k according to an enabling function.

$$\begin{cases} E_j=1,\ E_k=0 & \textit{if } |P_j|<|P_k| \\ E_j=0,\ E_k=1 & \textit{Otherwise} \end{cases}$$



e) **Finite buffer sizes and blocking** When $| P_{bf} | = K$, the token (customer) in P_i is blocked.



f) **Priority** By assigning higher priority to t_j other than t_k, the conflict is resolved.

Figure 3.2 Modelling Power of Petri Net Models

3.4 Stochastic Petri Net Package

The analysis of a GSPN model is made via its underlying Markov chain. When GSPN are used for the representation of real systems, these Markov chains tend to have a large state space, so that their numerical solution is computationally expensive and often even impossible. Thus, it becomes a challenging task for probabilists and applied mathematicians to develop solution methods and algorithms that allow the analysis to be performed (Ajmone 1988). The Stochastic Petri Net Package, Version 3.1, developed at Duke University is one of the available tools for solving GSPN (Ciardo *et al.* 1992). To get the performance results by SPNP, the only thing required is constructing the Petri net structure and specify parameters for places and transitions. The basic inputs and outputs are listed in **Table 3.2**.

Table 3.2 Basic Inputs and Outputs of SPNP

Inputs Required	Net Structure: • places • transitions • arcs Parameters & Functions: • initial marking • transition firing rates • transition priorities • transition enabling functions • transition switching functions
Outputs Obtained	Places: • non-empty probability • average # of tokens Transitions: • enabled probability • average throughput

Chapter 4

PETRI NET MODELS OF AN AUCTION-BASED MANUFACTURING CELL

4.1 Introduction

To investigate the heterarchical control strategy in an environment with multi-part-type and multi-machining-centre, a two-stage flexible manufacturing cell with heterarchical control is modelled and analyzed. In the following sections, a description of cell configuration and related assumptions are presented first. Then, the Petri net models for three cases: single part type with instage buffers, multiple part types with instage buffers, and multiple part types without instage buffers are constructed. Finally, performance measures are defined in terms of GSPN.

4.2 Problem Statement

The configuration of an Flexible Manufacturing Cell is shown in **Figure 4.1**. This is a two-stage production system. The first stage (denoted by 1 in the first index) consists of *m* parallel facilities, each producing the same family of parts; the second stage (denoted by 2 in the first index) consists of *n* parallel

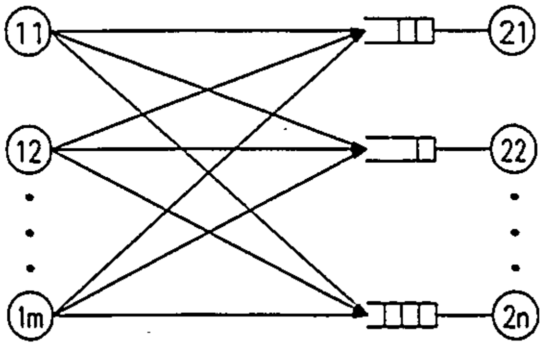


Figure 4.1 Configuration of an FMC

facilities, each being able to work on any part type from the first stage.

As a heterarchical control strategy is applied, no central controller (which is usually responsible for dispatching of parts from first stage to the second stage in a traditional Flexible Manufacturing Cell) exists between the two stages. A part from the upstream stage will select one of the downstream machines which offers the **Earliest Finishing Time** (EFT), i.e., the lowest expected time for processing on the machine and waiting in the buffer.

As a first instance, the input population is assumed to be infinite, i.e., there are always parts requesting service at the input of the system. A part will not enter the system until the upstream stage machine is free. Therefore, this implies a "pull" production control policy.

The processing time of a machine is assumed to be exponentially distributed. Define μ_{1jj}, $j=1,2,\dots,m$, to be the processing rate of upstream machine j for part type j, and μ_{2ij}, $i=1,2,\dots,n$, $j=1,2,\dots,m$, to be the processing rate of downstream machine i for part type j. Moreover, it is assumed that a machine can process only one part type at a time.

Instage buffers of various sizes K_i, $i=1,2,\dots n$, are available for each downstream machine to absorb the effect of imbalances between the production rates of successive stages. Parts in a buffer follow the First In First Out (FIFO) queue discipline.

To simplify the problem, it is assumed that the radio communication network has enough channel capacity to handle the bidding requests and responses, *i.e.*, the negotiation process. Transportation facilities are available whenever needed. The communication delays and transportation time are negligible.

A question may arise when there is more than one part among upstream machines waiting to enter the downstream stage, i.e., which part should go first. Before the question is answered, the first thing that should be clarified is that this situation may occur only when there are no downstream machines available for a part's request. Therefore, the part has to wait a back-off time before its next calling. During this time, a second part may come to request entry to the downstream stage. As all the parts are autonomous, the second part calls for service immediately without knowing that another part has already arrived and waiting for service, and perhaps it gets responses and enters one of the downstream buffers. Therefore, the part calling in first may not get into the downstream stage first. So in the event that more than one part is waiting to enter the downstream stage, the probability of acceptance is assumed to be equal for all such parts.

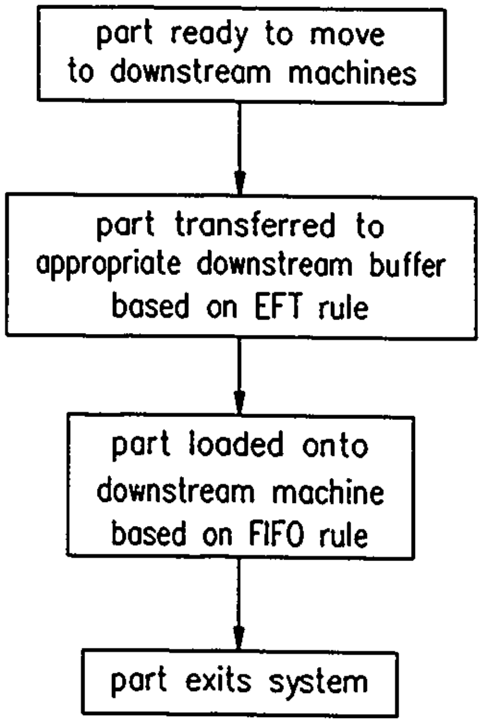

part ready to move to downstream machines
part transferred to appropriate downstream buffer based on EFT rule
part loaded onto downstream machine based on FIFO rule
part exits system

Figure 4.2 The Auction Process

The auction process modelled in this thesis is shown in **Figure 4.2**.

4.3 Petri Net Models

4.3.1 Model I: Single Part Type with Instage Buffers

A Petri net model for one part type, one upstream machine ($m=1$), n downstream machines, and instage buffers of size K_i, $i \in [1,n]$, is depicted in **Figure 4.3**. The related notations are given in **Table 4.1**.

Table 4.1 Notations for Figure 4.3

	Symbol	Physical Meaning
Places	P_{f1}	upstream machine free
	P_{i1}	upstream machine idle
	P_{bfik}	free place(s) at position k in buffer i, $i=1,2,....n$; $k=1,2,...,K_i$
	P_{ik}	part at position k in buffer i
	P_{b2i}	downstream machine i busy
	P_{f2i}	downstream machine i free
Transitions	t_{p1}	upstream machine processing a part
	t_{bf2i}	part entering buffer i
	t_{ik}	part in buffer i moved from position k+1 to k, $k=1,2,...,K_i$
	t_{m2i}	part removed from buffer i to downstream machine i
	t_{p2i}	downstream machine i processing a part

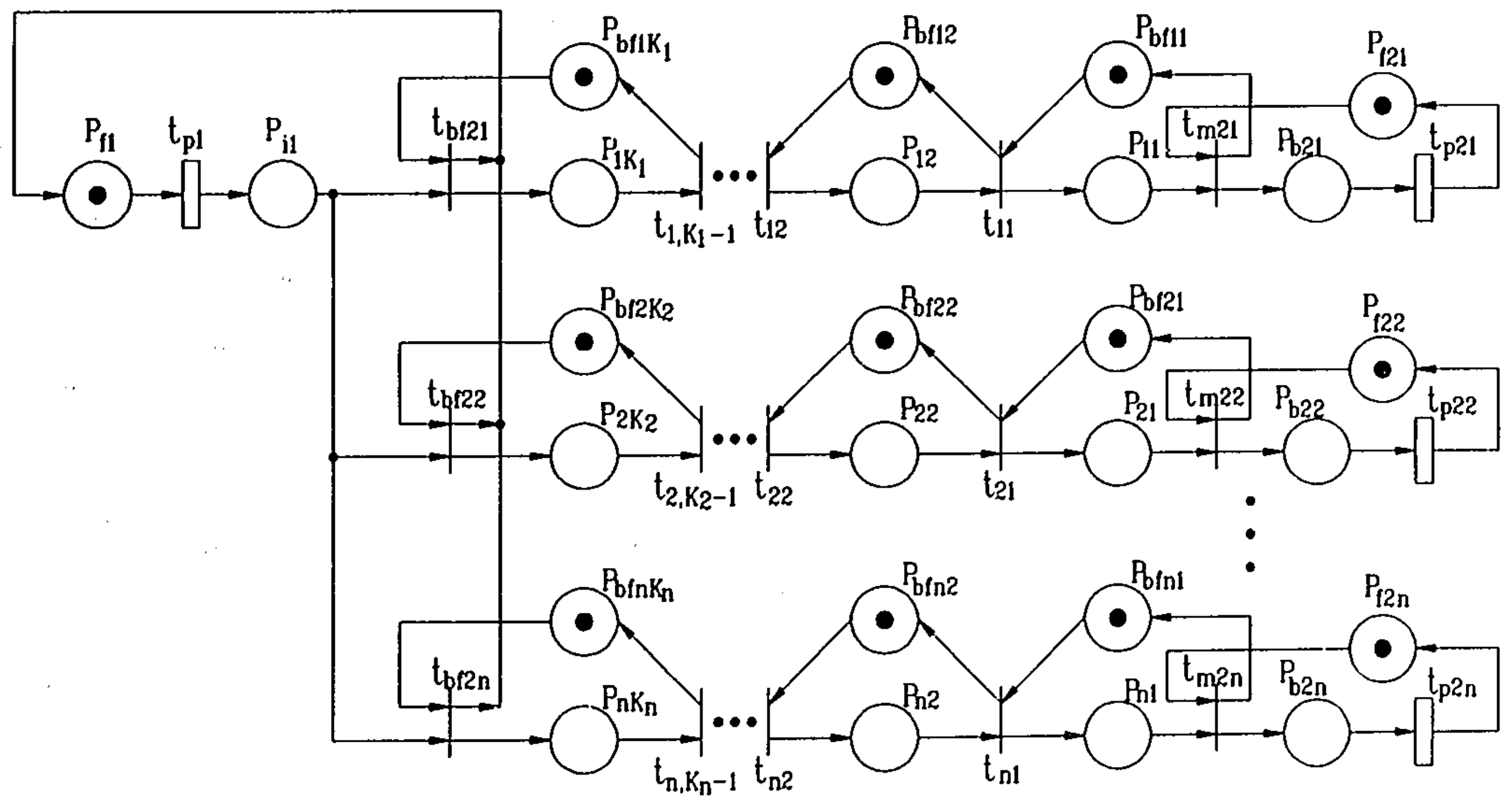


Figure 4.3 A Petri Net Model for an FMC with One Part Type

The local auction rule can be implemented by defining an enabling function for transition t_{bf2i}, $i \in [1,n]$. As is known, only when there are free places in the buffer, a downstream machine will respond to the requests from the upstream. So choices are made only among those downstream machines whose instage buffers are not full. Define

$$F = \{\, i : \sum_{k=1}^{K_i} |P_{bfik}| \neq 0,\ i = 1,2,\ldots,n \,\} \tag{1}$$

as the set of machines with free buffer place(s), where $|P_{bfik}|$ represents the number of token in place P_{bfik} (either 0 or 1). The time acknowledged by machine i is the expected processing time on a part and the average waiting time in buffer i, which is defined as the "*bid time*" of downstream machine i, denoted by BT(i). In terms of Petri nets, BT(i) can be expressed by the following equation:

$$\begin{aligned} BT(i) &= processing\ time + waiting\ time \\ &= \frac{1}{\mu_{2i}} + \frac{\sum_{k=1}^{K_i} |P_{ik}|}{\mu_{2i}} + \frac{|P_{b2i}|}{\mu_{2i}} \qquad \forall i \end{aligned} \tag{2}$$

where μ_{2i} is the mean processing rate of downstream machine i.

The winner, machine L, will be the one meeting the following conditions:

$$BT(L) = \min_{\forall i \in F} \{BT(i)\} \tag{3}$$

Therefore, t_{bf2i}, $\forall i$, can be enabled only when i=L. The enabling function of t_{bf2ij}, is

$$E(t_{bf2i}) = \begin{cases} 1 & if\ i = L \\ 0 & if\ i \neq L \end{cases} \qquad \forall i \tag{4}$$

To resolve the conflict of t_{ik}, $\forall i \in [1,n]$, equal probabilities are given to the enabled t_{ik}. The random switch is defined as

$$RS(t_{ik}) = \frac{1}{|H(t_{ik})|} \qquad \forall i \tag{5}$$

where $|H(t_{ik})|$ represents the number of enabled t_{ik}, $\forall i$, in a marking M.

Similarly, to resolve the conflict of t_{m2i}, the random switch is defined as

$$RS(t_{m2i}) = \frac{1}{|H(t_{m2i})|} \qquad \forall i \tag{6}$$

To resolve the conflict of transitions t_{bf2i}, t_{ik}, and t_{m2i}, $\forall(i,k)$, priorities associated with these transitions are defined as follows:

$$\begin{aligned} &priority(t_{bf2i}) = 1 && \forall i \\ &priority(t_{ik}) = K_i + 1 && \forall i \\ &priority(t_{m2i}) = K_i + 1 - k && \forall i \end{aligned} \tag{7}$$

In the following, the working process of the PN model is explained briefly. The initial marking indicates that the upstream machine (P_{f1}) is processing a part, all the downstream machines (P_{f2i}, i=1,2,...n) are free, and the instage buffers (P_{bfi}, i=1,2,...,n; j=1,2,..., K_i) are empty. Thus, the exponential transition t_{p1} is the only one enabled, and hence this is a tangible marking. After time $1/\mu_1$ has elapsed, t_{p1} fires, removing one token from P_{f1} to P_{i1}, which means that the upstream machine is idle and the finished part requests entry into one of the downstream machines. As all the instage buffers are empty, the part selects the "path" which gives the shortest finishing time, say, machine 2. As buffer 2 is empty, only immediate transition t_{bf22} is enabled, which is a vanishing marking. The firing of t_{bf22} removes the tokens in P_{bf2K_2} and P_{i1}, and deposits one token into P_{2K_2}. Firing of immediate transitions t_{2j}, j=K_2-1, ..., 1, consecutively results in a token in P_{21}, which means a part at position 1 in buffer 2 is ready to be loaded onto machine 2. As machine 2 is free (P_{f22}), transition t_{m22} is enabled and fires, removing tokens from P_{f22} and P_{21} and putting one token into P_{b22}, indicating that machine 2 is busy with a part (a tangible marking) and one token to P_{bf21}, showing that position 1 in buffer 2 is empty. After a processing time $1/\mu_{22}$, exponential transition t_{p22} finishes firing, removing the token from P_{b22} and assigning one to P_{f22}, which means the downstream machine 2 is free again.

4.3.2 Model II: Multiple Part Types with Instage Buffers

The Petri net model for the m part type case can be obtained by combining the structure in Figure 4.3, is shown in **Figure 4.4**. The associated notations are listed in **Table 4.2.**

Table 4.2 Notations for Figure 4.4

	Symbol	Physical Meaning
Places	P_{f1j}	upstream machine j free
	P_{i1j}	upstream machine j idle
	P_{bfik}	free place(s) at position k in buffer i, i=1,2,...,n; k=1,2,...,K_i
	P_{ijk}	part type j at position k in buffer i
	P_{b2ij}	downstream machine i busy with part type j
	P_{f2i}	downstream machine i free
Transitions	t_{p1j}	upstream machine j processing part type j
	t_{bf2ij}	part type j entering buffer i at position K_i
	t_{ijk}	part type j in buffer i moved from position k+1 to k
	t_{m2ij}	part type j removed from buffer i to downstream machine i
	t_{p2ij}	downstream machine i processing part type j

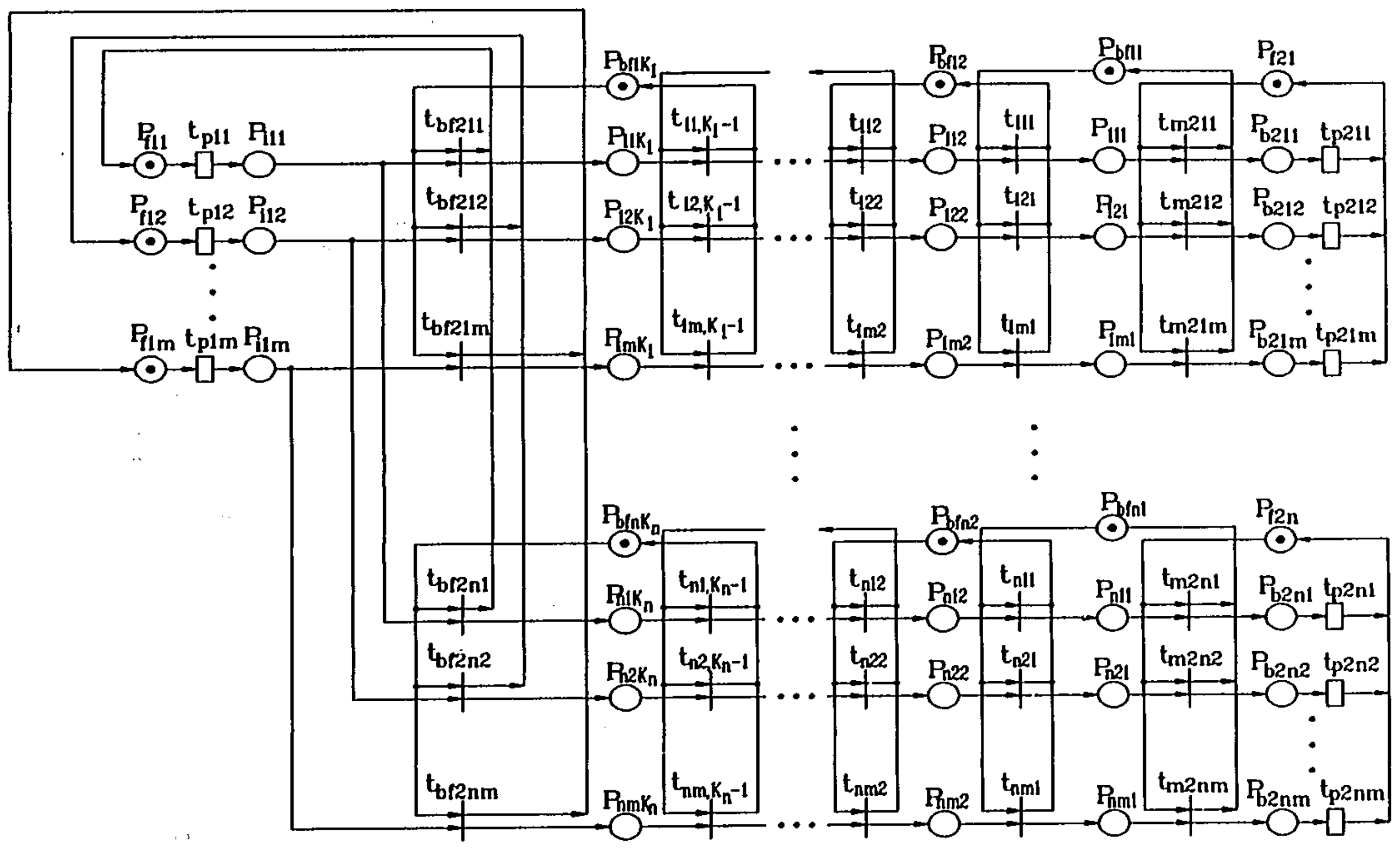


Figure 4.4 A Petri Net Model for an FMC with Multiple Part Types

To implement the local auction rule, the bid time of machine i for part type j, denoted as BT(i,j), can be expressed by the following equation:

$$BT(i,j) = processing\ time + waiting\ time$$
$$= \frac{1}{\mu_{2ij}} + \frac{\sum_{k=1}^{K_i} |P_{ijk}|}{\mu_{2ij}} + \frac{|P_{b2i}|}{\mu_{2ij}} \qquad \forall(i,j) \tag{8}$$

where μ_{2ij} represents the mean processing rate of downstream machine i for part type j. The winner, machine L, will be the one meeting the following conditions:

$$BT(L,j) = \min_{\forall i \in F} \{ BT(i,j) \} \qquad \forall j \tag{9}$$

where set F is defined in Equation (1). Therefore, t_{bf2ij}, $\forall$(i,j), can be enabled only when i=L. The enabling function of t_{bf2ij} is

$$E(t_{bf2ij}) = \begin{cases} 1 & if\ i = L \\ 0 & if\ i \neq L \end{cases} \qquad \forall(i,j) \tag{10}$$

As mentioned in section 4.1, when there is more than one part ready to enter the downstream stage, equal probabilities are assigned to each part. These events are described by enabled immediate transition t_{bf2ij}, $\forall$(i,j), denoted by H(t_{bf2ij}). The switching distribution, RS(t_{bf2ij}), is as follows.

$$RS(t_{bf2ij}) = \frac{1}{|H(t_{bf2ij})|} \qquad \forall(i,j) \tag{11}$$

where | H(t_{bf2ij}) | represents the number of enabled t_{bf2ij}, $\forall$(i,j), in a marking M.

To resolve the conflict of t_{ijk}, $\forall$(j,k), equal probabilities are given to the enabled t_{ijk}. The random switch is defined as

$$RS(t_{ijk}) = \frac{1}{|H(t_{ijk})|} \qquad \forall(i,j) \tag{12}$$

Similarly, the random switch for t_{m2ij}, $\forall$(i,j), is defined as follows to resolve the conflict of t_{m2ij}.

$$RS(t_{m2ij}) = \frac{1}{|H(t_{m2ij})|} \qquad \forall(i,j) \tag{13}$$

To resolve the conflict of transitions t_{bf2ij}, t_{ijk}, and t_{m2ij}, $\forall$(i,j), priorities associated with these transitions are defined as follows.

$$\begin{array}{ll} priority(t_{bf2ij}) = 1 & \forall(i,j) \\ priority(t_{ijk}) = K_i + 1 & \forall(i,j) \\ priority(t_{m2ij}) = K_i + 1 - k & \forall(i,j) \end{array} \tag{14}$$

4.3.3 Model III: Multiple Part Types without Instage Buffers

It should be noted that the Petri net model in Figure 4.4 is valid only when the buffer size $K_i > 0$, $\forall$i. If no instage buffer exists in front of each downstream machine, the Petri net structure is as shown in **Figure 4.5**. The related notations are the same as those for Figure 4.4, except that t_{m2ij} represents the event that part type j moves from upstream machine j to downstream machine i.

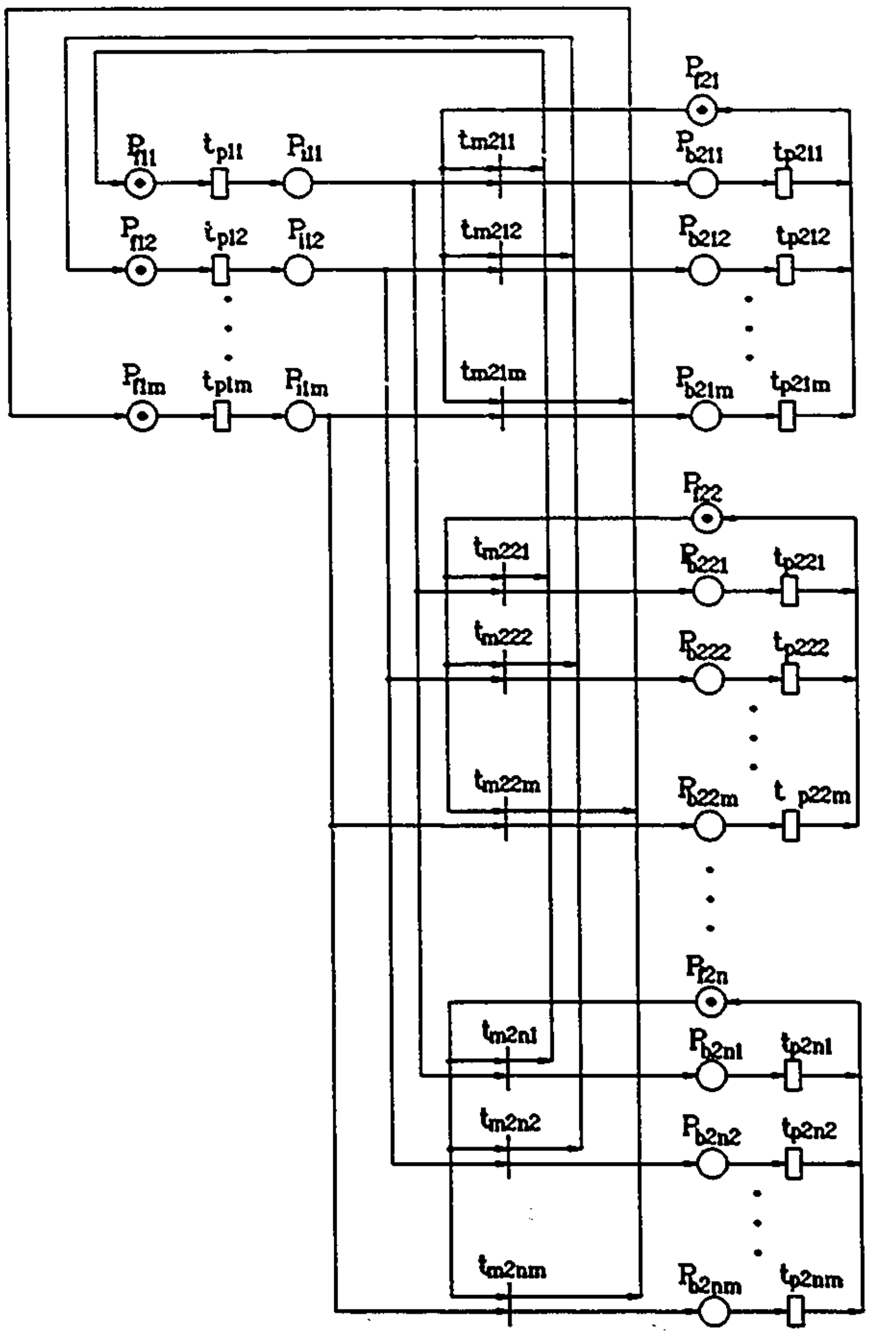


Figure 4.5 A Petri Net Model for an FMC Without Instage Buffers

4.4 Performance Measures

1) Arrival Intensity

To describe the relative speed of the arrival process of parts from the upstream to downstream stage, *arrival intensity*, λ, is defined as the ratio of total mean processing rates of upstream machines to the total of the average processing rates of downstream machines.

$$\lambda = \frac{\sum_{j=1}^{m} \mu_{1j}}{\sum_{i=1}^{n} (\frac{1}{m} \sum_{j=1}^{m} \mu_{2ij})} = \frac{m \sum_{j=1}^{m} \mu_{1j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} \mu_{2ij}} \tag{15}$$

2) Machine Utilization

For upstream machine j, machine utilization, UT1(j), is the probability with that machine j is busy. In terms of the Petri net model, it is the probability with that transition t_{p1j} is enabled, i.e.,

$$UT1(j) = Pr\{t_{p1j} \ enabled\} \quad \forall j \tag{16}$$

Similarly, for downstream machine i, the *overall* utilization is the probability with that machine i is busy with all the part types. Define the *detailed* utilization of machine i is the probability of machine i busy with part type j, UT2(i,j), *i.e.*,

$$UT2(i,j) = Pr\{t_{p2ij} \ enabled\} \qquad \forall (i,j) \tag{17}$$

The *overall* utilization of downstream machine i, UT2(i), is

$$UT2(i) = \sum_{j=1}^{m} UT2(i,j) = \sum_{j=1}^{m} Pr\{t_{p2ij} \ enabled\} \qquad \forall i \tag{18}$$

3) Machine Throughput

The *average throughput* E[t_θ] for transition θ is defined as (Ciardo 1992)

$$TH(t_\theta) = \sum_{i \in R(\theta)} p(M_i) * \mu(\theta, M_i) \tag{19}$$

where R(θ) is the subset of reachable markings that enable transition θ, p(M_i) is the probability of marking M_i, and $\mu(\theta, M_i)$ is the rate of transition θ in marking M_i.

For an exponential transition with a non-marking dependent firing rate, the *average throughput* is the product of the probability with which the transition is enabled, p(θ), and its firing rate, μ_θ. According to the definition of utilization, we have

$$TH(t_\theta) = p(\theta) * \mu_\theta = UT(\theta) * \mu_\theta \tag{20}$$

For a manufacturing system, throughput represents the number of parts produced per unit time (per hour or day). For upstream machine j, the average throughput, TH1(j), is the throughput of transition t_{p1j}, i.e.,

$$TH1(j) = throughput(t_{p1j}) \qquad \forall j \tag{21}$$

For downstream machine i, the average throughput for part type j (detailed throughput), TH2(i,j), is

$$TH2(i,j) = throughput(t_{p2ij}) \qquad \forall (i,j) \tag{22}$$

The average throughput of downstream machine i for all part types (overall throughput), TH2(i), is

$$TH2(i) = \sum_{j=1}^{m} TH2(i,j) = \sum_{j=1}^{m} throughput(t_{p2ij}) \qquad \forall i \tag{23}$$

4) Number of Blocked Jobs in Upstream Machine j

The average number of blocked jobs in upstream machine j, BQL(j), j=1,2,...,m, is defined as the average number of tokens in place P_{i1j}, j=1,2,...,m, *i.e.*,

$$BQL(j) = |P_{i1j}| \qquad \forall j \tag{24}$$

The average number of blocked jobs in upstream machines is

$$BQL = \sum_{j=1}^{m} QL(j) = \sum_{j=1}^{m} |P_{i1j}| \tag{25}$$

5) Queue Length in Buffer i

The average queue length in buffer i, QL(i), i=1,2,...,n, can be obtained by the following equation.

$$QL(i) = K_i - \sum_{j=1}^{j=m} |P_{bfij}| \qquad \forall i \tag{26}$$

The average queue length in downstream buffers is

$$QL = \sum_{i=1}^{n} QL(i) \tag{27}$$

Chapter 5

EXAMPLES AND RESULTS

5.1 Introduction

Two examples are given in this chapter, which show how the performance measures can be obtained by the developed Petri net models using the SPNP package. The purpose of the first example is to verify, somehow, the developed model by comparing the results with what has been reported. The second example serves to indicate the behaviour of the auction-based manufacturing cell. Some observations and discussions are presented based on the results.

5.2 Example I: Single Part Type

Some results have been reported about the behaviour of an auction-based manufacturing system with a single part type (Upton, 1991). In this section, Petri Net Model I (single part type) is applied to analyze an example problem with one part type and six downstream machines, to see if the results obtained by the model are consistent with those reported, which in turn verifies the feasibility of the developed Petri net model. The parameters for the example, i.e., mean processing rates of machines and sizes of instage buffer, are provided in **Table 5.1**.

In the following, the performances of upstream machines and downstream machines will be discussed respectively.

Table 5.1 Parameters for Example I

Mean Processing Rate (μ_{ij})		Buffer Size (K_i)
Upstream Machine 1	2.1 - 42	-
Downstream Machine 1	6.0	1
Downstream Machine 2	5.0	1
Downstream Machine 3	4.0	1
Downstream Machine 4	3.0	1
Downstream Machine 5	2.0	1
Downstream Machine 6	1.0	1

Performance of Upstream Machines:

The throughput, utilization, and the number of blocked jobs of upstream machines are given in **Figures 5.1a-c**. The following results can be obtained by examing these curves.

- When the arrival intensity is low, the second stage can handle all the requests from the upstream stage. Therefore, very few jobs are blocked in the upstream machine and upstream machine utilization is high. However, the throughput is low due to the low processing rate.
- With the increase of the arrival intensity, the system gets more heavily loaded, which results in an increased number of blocked jobs, decreased machine utilization, and increased throughput.
- Finally, the system reaches a saturated state because of the limited capacity of the downstream machines and their instage buffers. Thus, the throughput tends to reach a constant level ($\Sigma_j\mu_{2j}$) and the utilization decreases dramatically because more finished parts are blocked from the upstream stage.

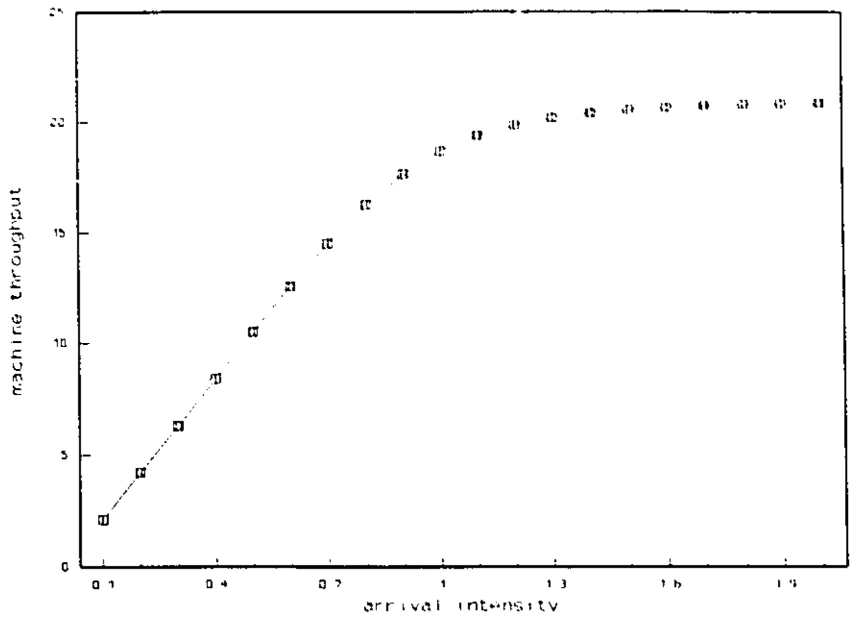


Figure 5.1a Throughput of the upstream machine when $m=1$, $n=6$, and $K_i=1$, $\forall i \in [1,n]$.

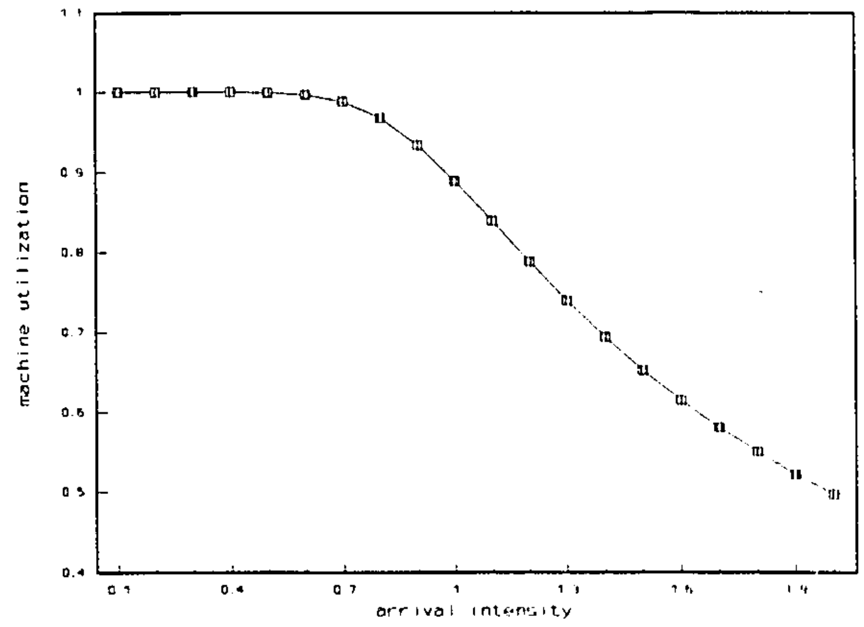


Figure 5.1b Utilization of the upstream machine when $m=1$, $n=6$, and $K_i=1$, $\forall i \in [1,n]$.

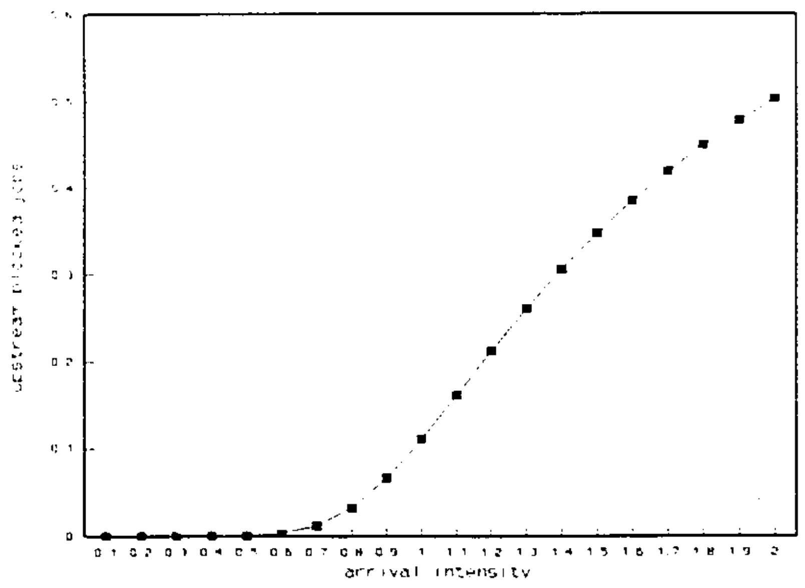


Figure 5.1c Number of blocked jobs of the upstream machine when m=1, n=6, and $K_i=1$, $\forall i \in [1,n]$.

Performance of Downstream Machines:

Figures 5.2a-c provide the throughputs, utilizations, and the queue lengths in the buffers of the downstream machines. The following observations can be made from these figures.

- In a very idle situation, the best machine is likely to win all of the requests from the upstream stage as a result of applying the EFT rule. It obtains the highest utilization and throughput, with the longest queue built up in its instage buffer.
- As the system becomes busier, slower machines gradually take on more work. Their utilizations and throughputs get higher compared with the idle case. The queue lengths of slower machines are built up more quickly than the faster ones. This all takes place because a part from the upstream stage has to accept the downstream machine even though it may not be the most effective one.

- The system reaches a saturation point when it gets extremely busy: all downstream machines are working at their full capacity (i.e., utilization is about 1). Therefore, the throughput of each downstream machine is determined by its processing rate, i.e., it receives jobs roughly in proportion to its mean processing rate.

These results are similar to the conclusions reached by Upton (1991), which, to some extent, show that the Petri net model developed is working properly.

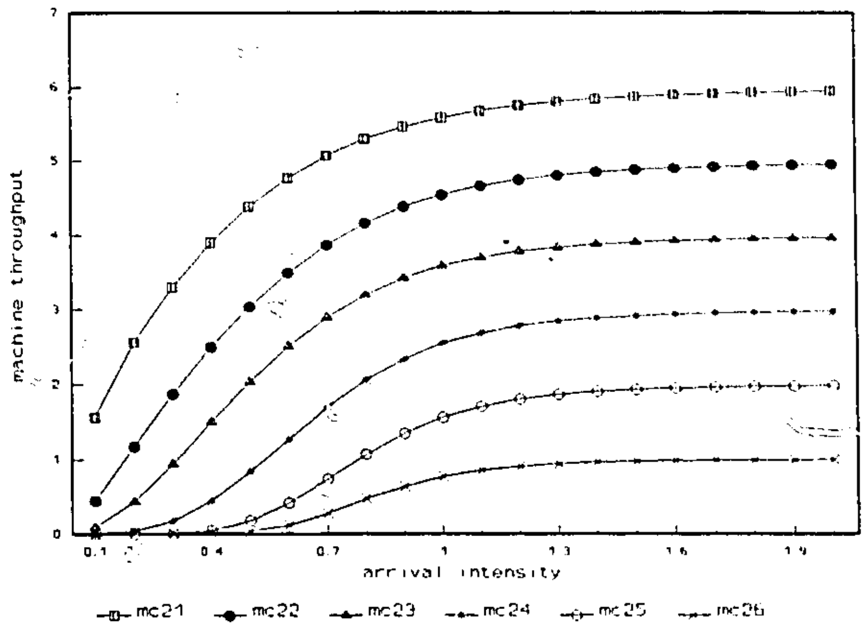


Figure 5.2a Throughputs of the downstream machines when m=1, n=6, and $K_i=1$, $\forall i \in [1,n]$.

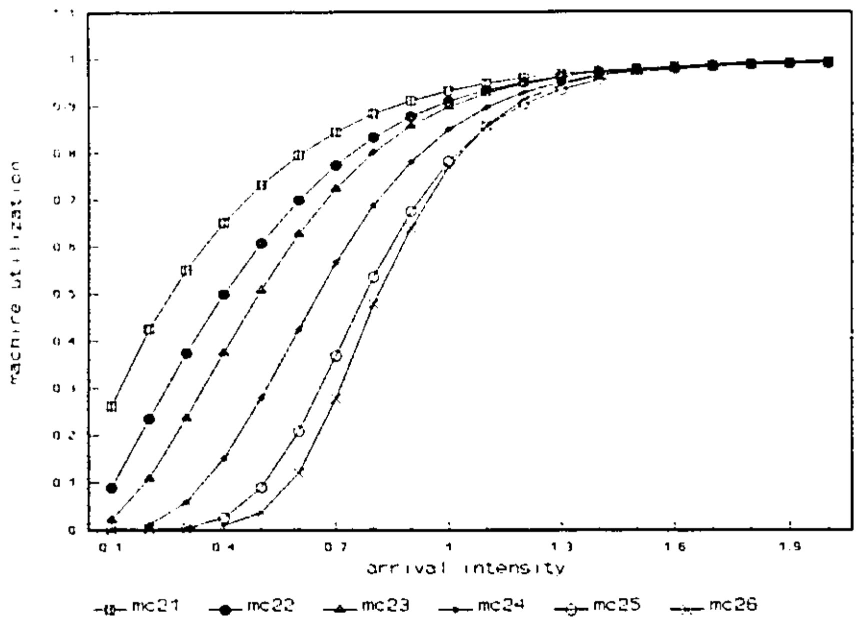


Figure 5.2b Utilizations of the downstream machines when m=1, n=6, and $K_i=1$, $\forall i \in [1,n]$.

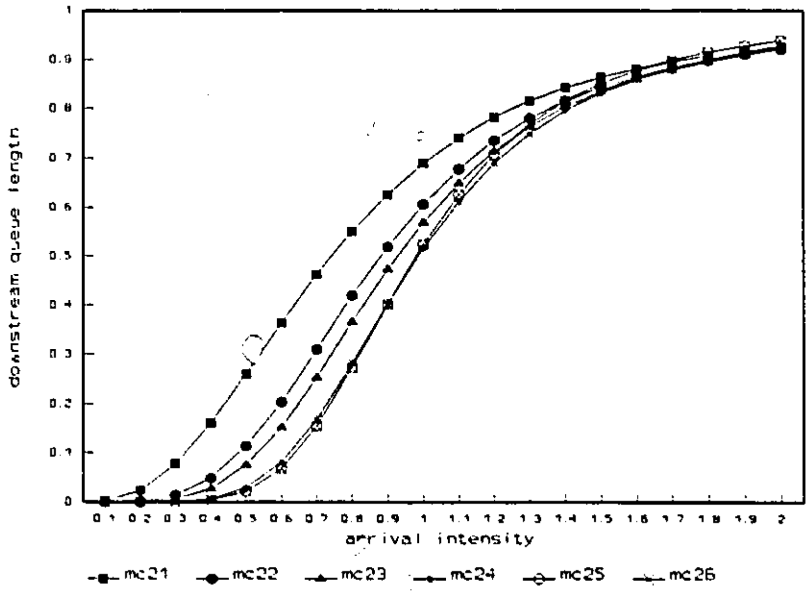


Figure 5.2c Queue lengths of instage buffers when m=1, n=6, and $K_i=1$, $\forall i \in [1,n]$.

5.3 Example II: Multiple Part Types

As the first step in investigating the behaviour of a heterarchical controlled system with multi-part-type and multi-machining-centre, consider a manufacturing cell with four upstream machines (m=4) and two downstream machines (n=2). The mean processing rates of machines are listed in **Table 5.2**. By this example, the following questions will be answered: first, how the part types from the upstream machines are allocated among the downstream machines; then, how the performance is influenced by the buffer sizes; finally, what the effect of number of part types on the performance measures is.

Table 5.2 Machining Centre Processing Rates for Example II

Part Type	Mean Processing Rate			
	Upstream Machine (unit / unit time)		Downstream Machine (unit / unit time)	
1	Machine 1	0.45 - 9.0	Machine 1	10.0
			Machine 2	8.0
2	Machine 2	0.35 - 7.0	Machine 1	5.0
			Machine 2	9.0
3	Machine 3	0.25 - 5.0	Machine 1	6.0
			Machine 2	4.0
4	Machine 4	0.15 - 3.0	Machine 1	2.0
			Machine 2	4.0

5.3.1 Auction Behaviour of Downstream Machines

As the allocation of jobs from the upstream stage among the downstream machines is the main issue in this section, a cell without instage buffers ($K_i=0$) is considered. PN Model III is used to evaluate the performance. To help in understanding the performance of downstream machines, the performance of upstream machines is briefly described first.

Performance of Upstream Machines:

Performance results are plotted in **Figures 5.3a-c**. It is observed that faster machines have higher throughputs, longer blocked queues because of the limited capacity of downstream facilities, and thus, lower utilizations.

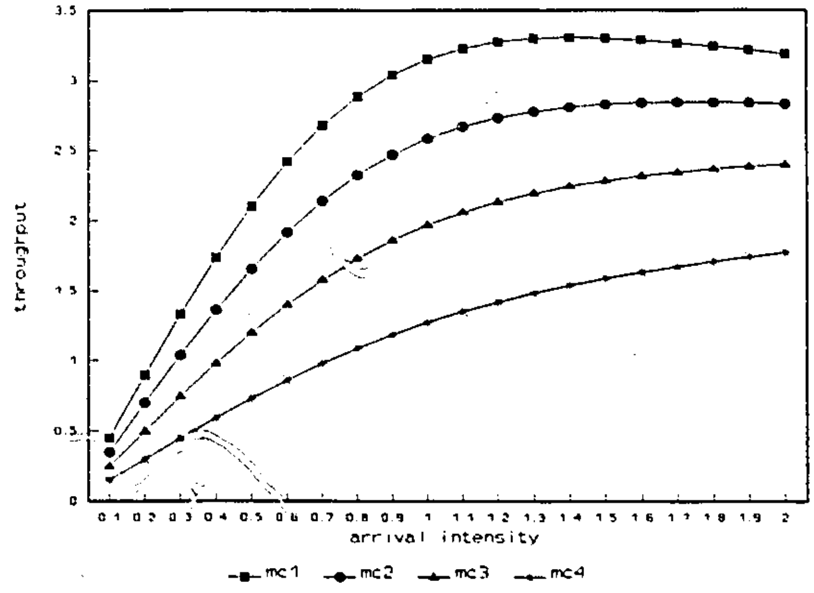


Figure 5.3a Throughputs of upstream machines when m=4, n=2, and K_i=0, $\forall i \in [1,n]$.

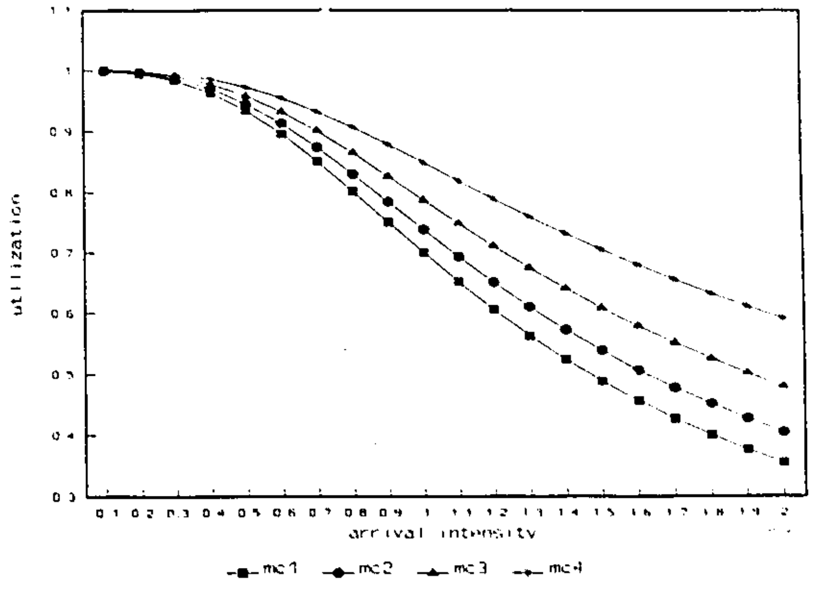


Figure 5.3b Utilizations of upstream machines when $m=4$, $n=2$, and $K_i=0$, $\forall i \in [1,n]$.

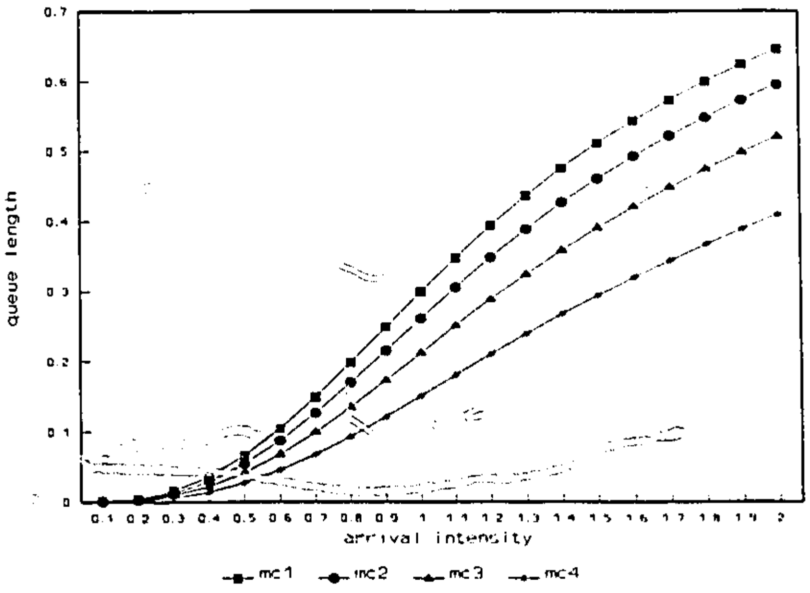


Figure 5.3c Number of blocked jobs of upstream machines when $m=4$, $n=2$, and $K_i=0$, $\forall i \in [1,n]$.

Performance of Downstream Machines:

As each downstream machine can work on four part types, its throughput and utilization has four components, too. **Figures 5.4a-h** give the throughputs and utilizations *with respect to each of the four part types.*

It is interesting to notice that the allocation of one part type from the upstream is not always according to the processing rates of downstream machines with respect to this part type. In other words, faster machines will not always have higher throughputs. This phenomena is different from that of one part type case. The following observations are made to see the details.

Figures 5.4a&b show the throughputs and utilizations of the two downstream machines with respect to *part type 1.* In terms of mean processing rates, machine 1 is faster than machine 2 for part type 1.

- **Utilization:** When $\lambda<0.9$, MC 1 with higher processing rate to part type 1 gets higher utilization. However, when $\lambda>0.9$, the slower MC 2 catches up and surpasses MC 1 in utilization. In fact, when the system is not busy, a part from the upstream has more chances to find the best downstream machine for its next operation. So the faster machine tends to have higher utilization. However, when the system is very busy, there are less possibilities that more than one downstream machine will be free at the same time. Most of the time, the part has to go to the only downstream machine which is free at that moment, in spite of the fact that the machine may not be the most effective one for the operation. Therefore, the utilization of the slower machine gets higher.

- **Throughput:** Although the utilizations of MC 1 and MC 2 switch at $\lambda=0.9$, their utilizations do not switch until $\lambda>1.3$. As mentioned in 4.2.2, the average machine throughput is the product of utilization and mean processing rate. Machine 2 has higher utilization when $\lambda>0.9$, but it is not high enough to overwhelm its lower processing rate until $\lambda>1.3$.

The results for *part type 2* are provided in **Figures 5.4c-d.** Here machine 2 is faster.

- **Utilization:** When $\lambda<0.8$, the utilization of the faster MC 2 is higher. However, when $\lambda>0.8$, the slower MC 1 has higher utilization. This phenomena is similar to that of part type 1.
- **Throughput:** Although the utilizations of MC 1 and MC 2 switch at $\lambda=0.8$, their throughputs do not switch when $0.8<\lambda<2.0$, because the utilization of the slower MC 1 is never big enough to overwhelm its lower processing rate. This situation is quite different from that of part type 1.

For *part type 3 and 4*, the results, shown in **Figure 5.4e-h**, are similar to that of part 1 and 2, respectively.

In summary, for an auction-based manufacturing cell with the "least finishing time" dispatching rule, the following conclusions can be reached:

- When the system is not so busy, the faster machine, with respect to a part type, has higher throughput than the slower one.
- As the system becomes very busy, the throughput of the faster machine for a particular part type may not be higher than the slower one, depending on the system

parameters, such as the combination of mean processing rates of downstream machines for different part types.

As mentioned above, when a system is very busy or saturated, a part from the upstream has little chance to choose the most effective downstream machine because almost all the downstream machines are occupied, and the part has to go to the only machine available at the time its request is broadcasted. For a particular part type, a faster machine may not get more jobs than a slower one. Hence, the faster one may not have higher throughput, which means that the advantage of the faster machine on a particular part type is not given full play. This is not what is expected and is a waste of resources. Thus, working around the saturation state should be avoided.

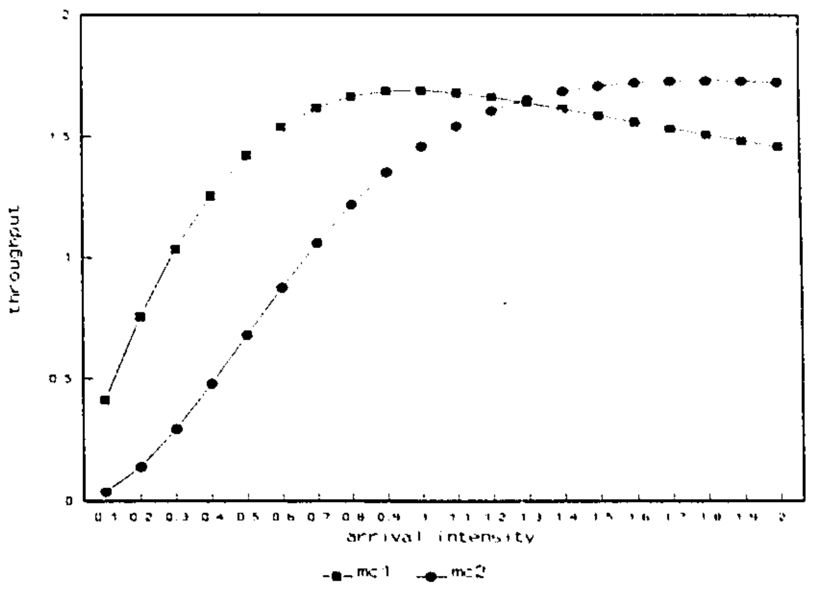


Figure 5.4a Throughputs of downstream machines for *part type 1* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

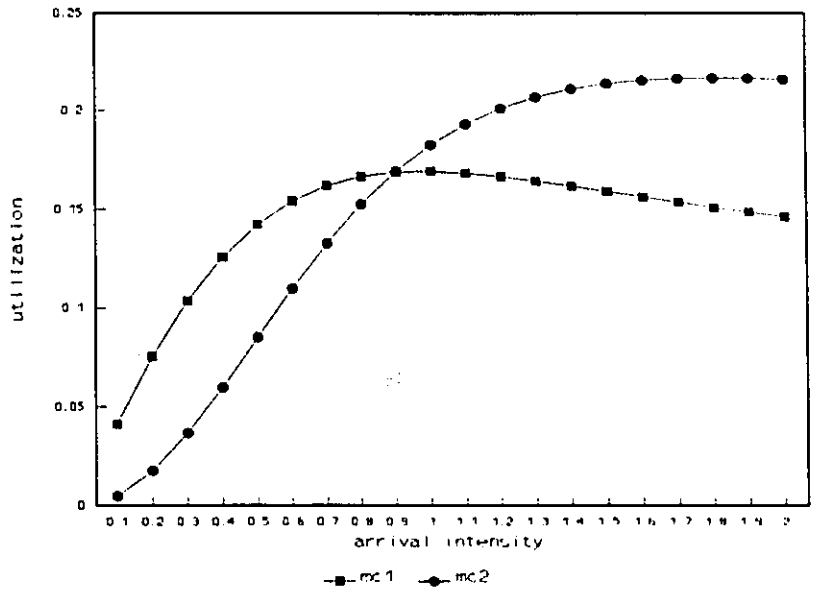


Figure 5.4b Utilizations of downstream machines for *part type 1* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

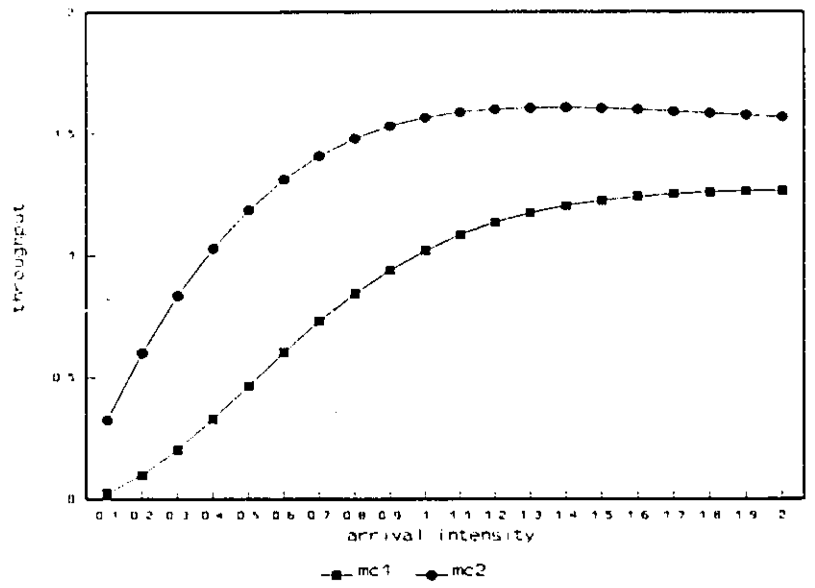


Figure 5.4c Throughputs of downstream machines for *part type* 2 when m=4, n=2, and K_i=0, ∀i∈[1,n].

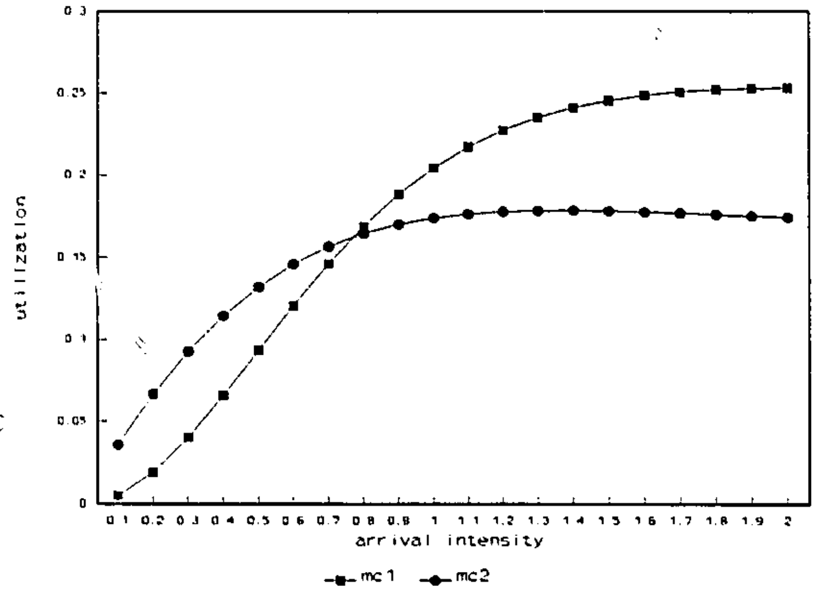


Figure 5.4d Utilizations of downstream machines for *part type* 2 when m=4, n=2, and K_i=0, ∀i∈[1,n].

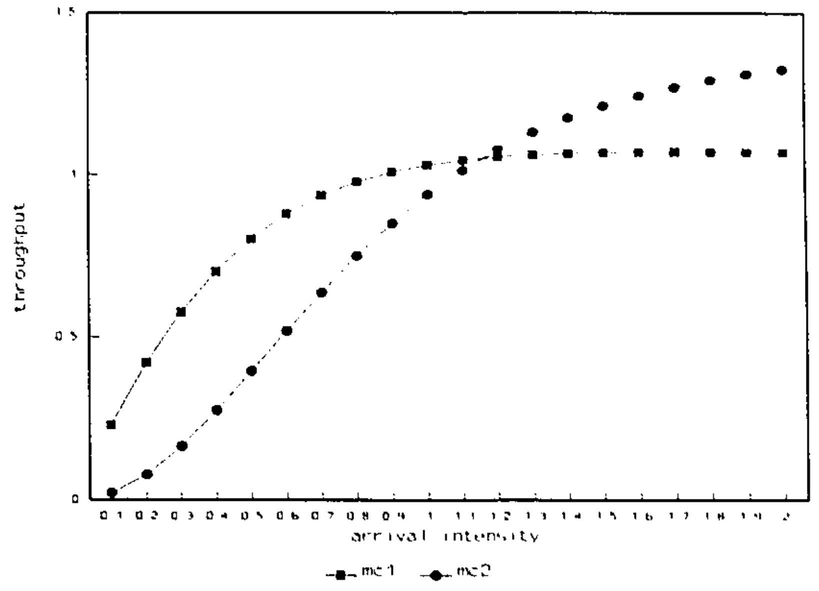


Figure 5.4e Throughputs of downstream machines for *part type 3* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

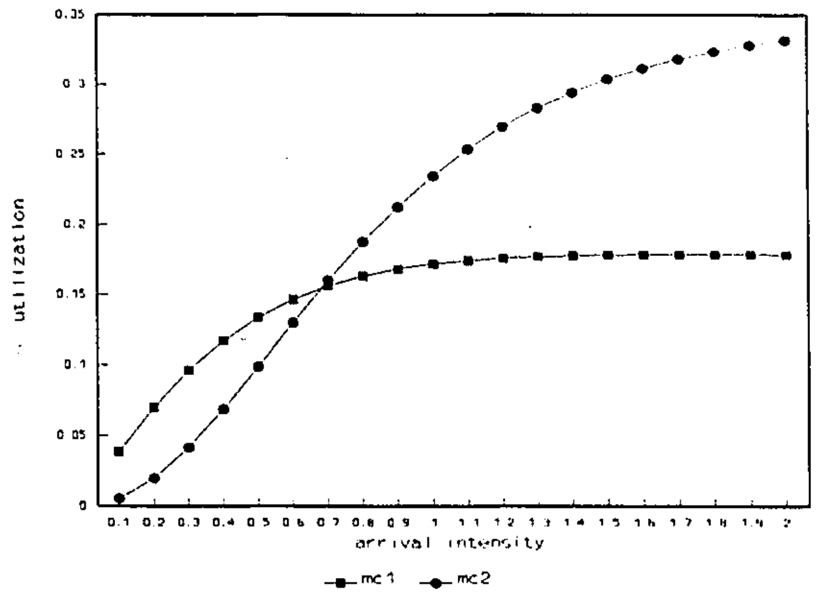


Figure 5.4f Utilizations of downstream machines for *part type 3* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

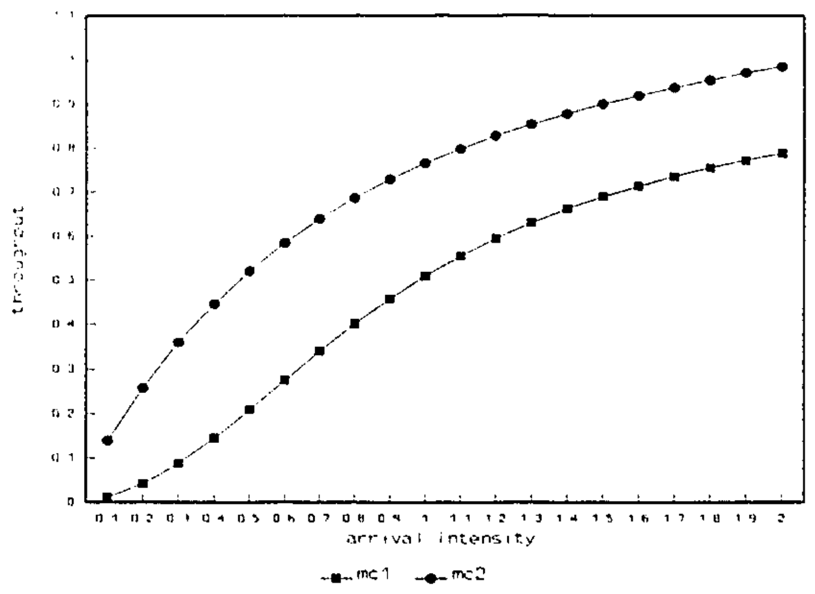


Figure 5.4g Throughputs of downstream machines for *part type 4* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

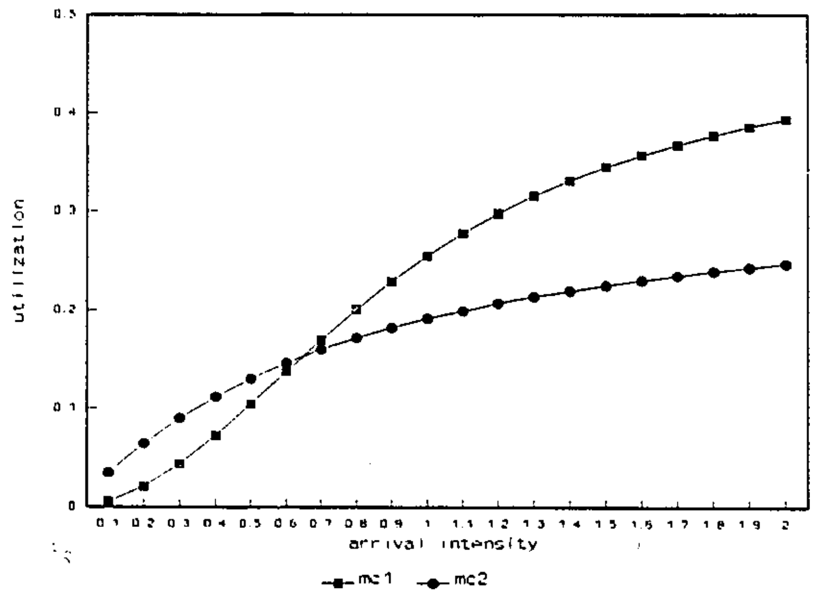


Figure 5.4h Utilizations of downstream machines for *part type 4* when m=4, n=2, and $K_i=0$, $\forall i \in [1,n]$.

5.3.2 Influence of Instage Buffer

To see the influence of buffer sizes, three cases are discussed and compared: without buffers, with a buffer of size 1, and with a buffer of size 2 . PN model II and III are applied to analyze the performance of both upstream machines and downstream machines.

Performance of Upstream Machines:

The typical results of upstream machines are illustrated by the performance measures of upstream machine 1, as shown in **Figures 5.5a-c.** The observations can be made as follows:

- When the system is idle, little difference exists between the three cases because all parts from the upstream stage can be processed by the downstream machines immediately. So in this case, it is not necessary for a machine to have an instage buffer.
- When the system is getting busier, higher throughputs and utilizations of upstream machines are obtained with the addition of an instage buffer.
- When the system is saturated, the performance with one or two buffers degrades gradually to the situation without buffers because downstream machines cannot handle all the service requests in time, even with the aid of buffers.

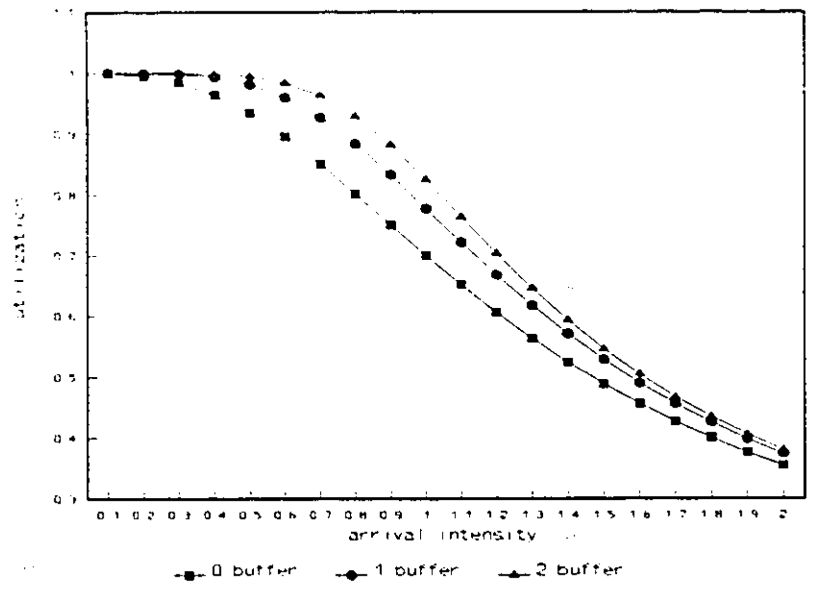


Figure 5.5a Utilizations of upstream machine 1 when m=4, n=2, and K_i=0,1,2, $\forall i \in [1,n]$.

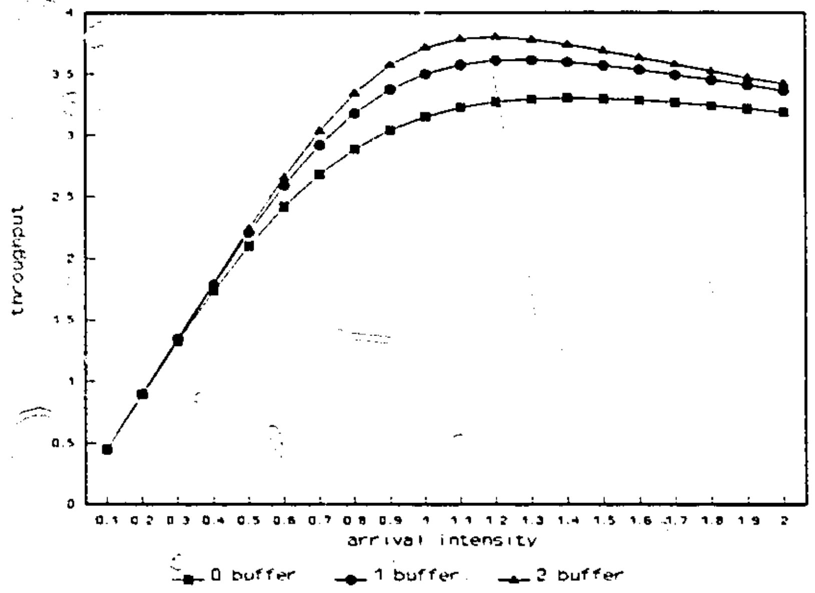


Figure 5.5b Throughputs of upstream machine 1 when m=4, n=2, and K_i=0,1,2, $\forall i \in [1,n]$.

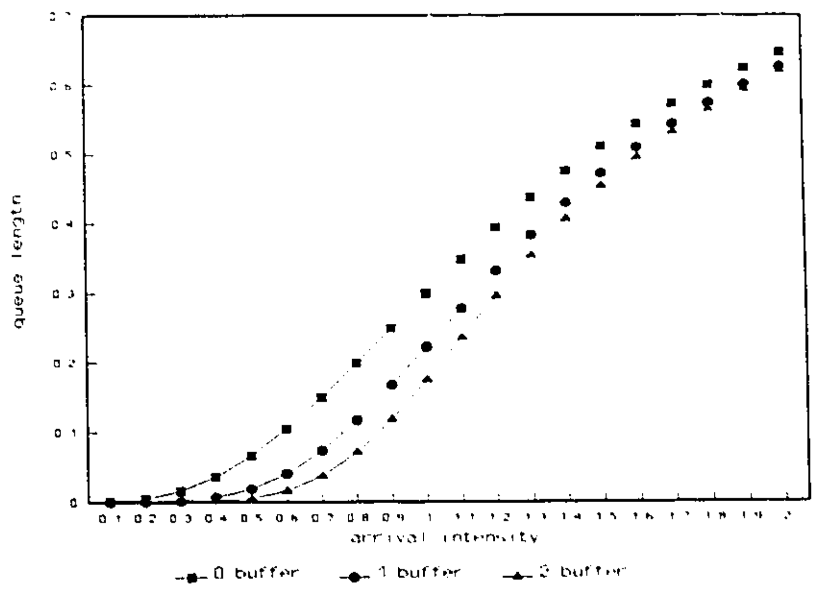


Figure 5.5c Number of blocked jobs of upstream machine 1 when m=4, n=2, and K_i=0,1,2, $\forall i \in [1,n]$.

Performance of Downstream Machines:

In terms of the *overall* **throughput** of a downstream machine (the throughput of one machine for all part types), which is given in **Figures 5.6a&b**, the observations are quite similar to that of upstream machines, *i.e.*, buffers bring benefits only when the system is neither idle nor saturated.

To see how the improvement is achieved, the *detailed* **throughput** of a downstream machine (the throughput of each downstream machine for each part type) is investigated and plotted in **Figures 5.7a-h**. The observations are as follows:

- When the system is not saturated, the throughput of the faster machine with respect to a particular part type can be improved by increasing the buffer size, but the throughput of the slower one is decreased.

- When the system approaches its saturation state, the throughputs of both faster and slower machines are improved.
- When the system is extremely saturated, little improvement in throughputs is achieved.

For example, **Figure 5.7a&b** shows the throughputs of downstream machine 1 and 2 with respect to part type 1. Here, machine 1 is faster than machine 2 for part type 1. When the system is not saturated ($\lambda < 0.9$), the throughput of machine 1 is improved by increasing the buffer sizes, but the throughput of machine 2 is degraded. When the system continues to become busier ($\lambda > 0.9$), the throughputs of machine 1 and 2 are both improved by larger buffer sizes. As λ approaches 2.0, the throughputs with buffer sizes 0,1,and 2 are almost the same. Similar situations can be seen from **Figure 5.7c-h.**

This can be explained as follows:

- When the system is not busy, a system with larger buffer sizes is freer compared with a cell with smaller buffer sizes or no buffers under a given arrival intensity. Therefore, a part from the upstream may have more chances to find the best machine. In other words, increase in buffer sizes will only improve the throughput of the faster machine with respect to a particular part type.
- However, when the system is so busy (around saturation point) that the local optimization process hardly exists (no choices), increasing buffer sizes could possibly create some chances for the system to have more than one machine free at the same time, which means that the local optimization process could happen. As

a result, the throughputs of downstream machines might be improved.

- As the system is highly saturated, increasing buffer sizes to 2 is not enough to handle the requests from the upstream stage. Hence, almost no improvement can be achieved.

In conclusion, increasing buffer sizes creates more chances for a part to find a better machine for its next operation, which results in the improvement in the overall throughput.

In general, the following conclusions can be reached:

1) When a system is not busy, it is not necessary to introduce instage buffers because they make no difference to the system performance.
2) When a system is highly saturated, there is little improvement on system performance with the addition of buffers of small sizes. On the other hand, buffers of large sizes are usually not practical. Therefore, there is no point in alleviating the saturation by adding buffers. Other measurements have to be taken, such as changing the criterion for scheduling, reducing the arrival intensity, adding more (updating) machining centres, *etc.*
3) When a system is between idle and lightly saturated, the system performance could be improved by increasing buffer sizes. However, the improvement may not be marginal as evident from Figure 5.6a-d. Therefore, before buffers are introduced into the system, cost analysis should be conducted regarding the benefit cost ratio with respect to increase in throughput vs. cost of adding and maintaining buffers in a real world situation.

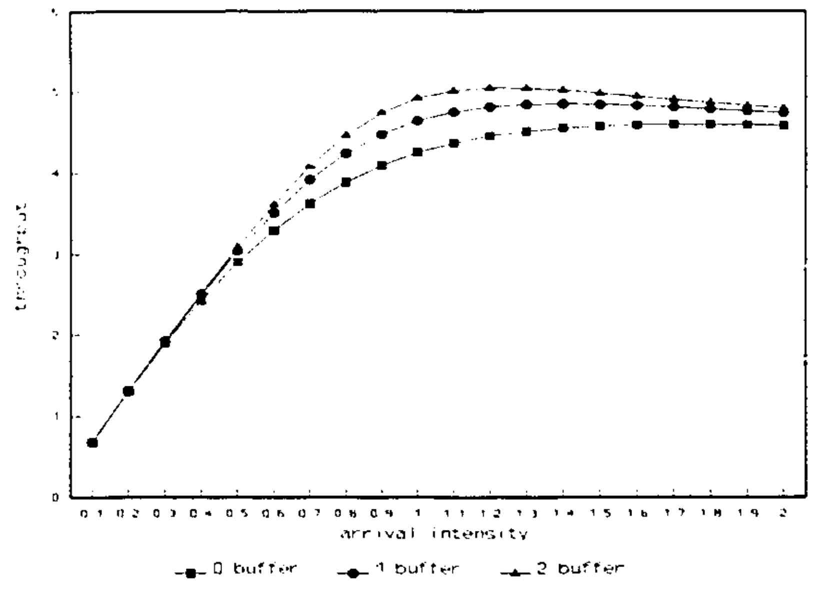


Figure 5.6a Throughputs of downstream machine 1 when m=4, n=2, and K_i=0,1,2, ∀i∈[1,n].

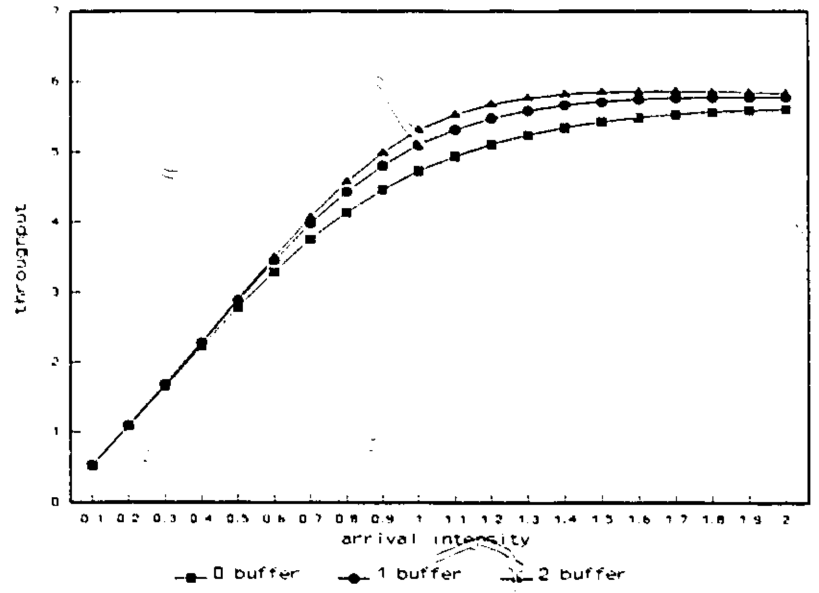


Figure 5.6b Throughputs of downstream machine 2 when m=4, n=2, and K_i=0,1,2, ∀i∈[1,n].

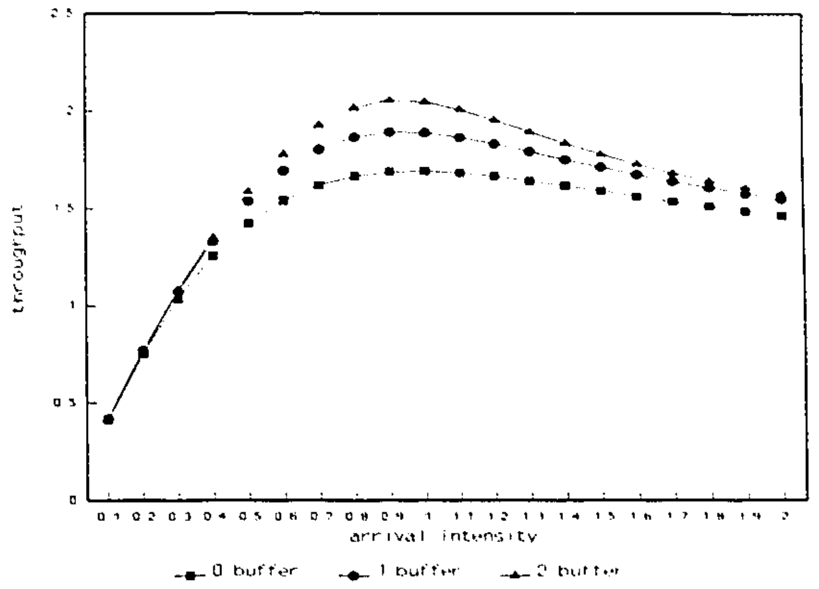


Figure 5.7a Throughputs of downstream machine 1 for *part type 1* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

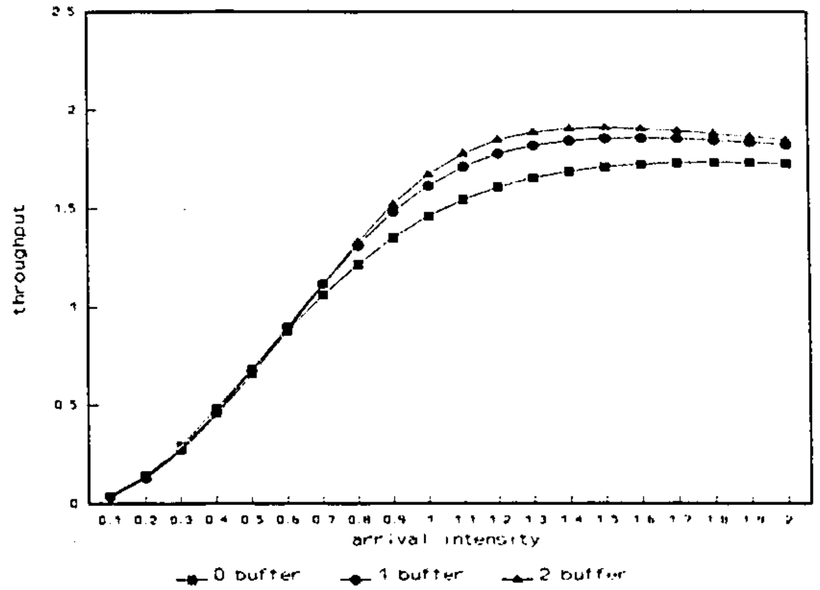


Figure 5.7b Throughputs of downstream machine 2 for *part type 1* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

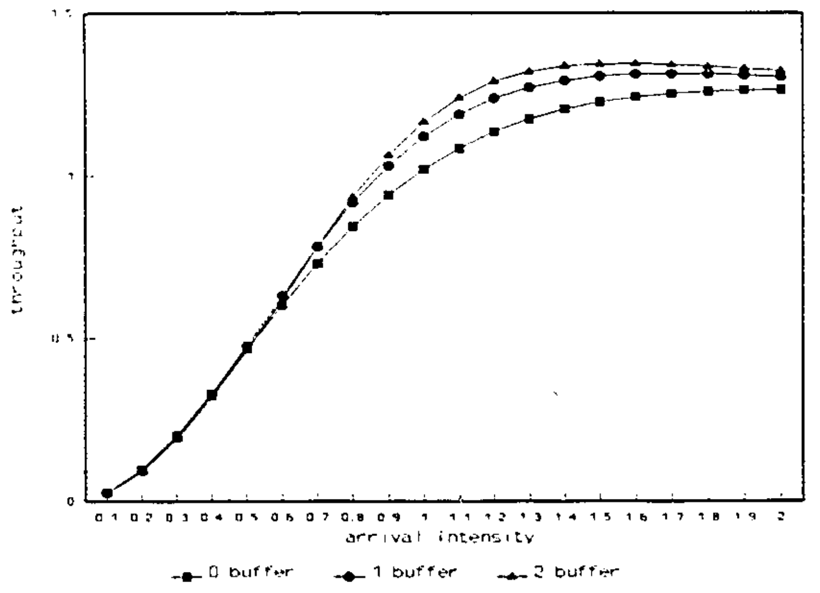


Figure 5.7c Throughputs of downstream machine 1 for *part type 2* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

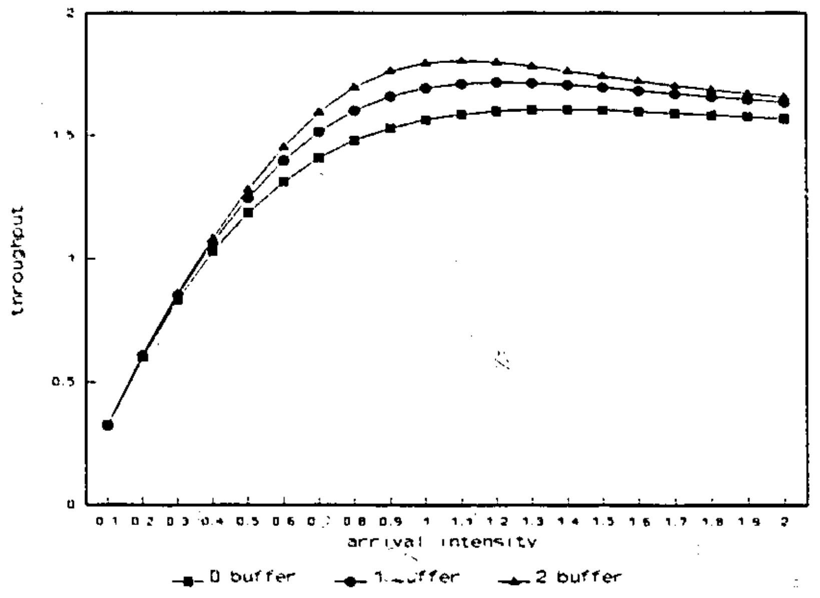


Figure 5.7d Throughputs of downstream machine 2 for *part type 2* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

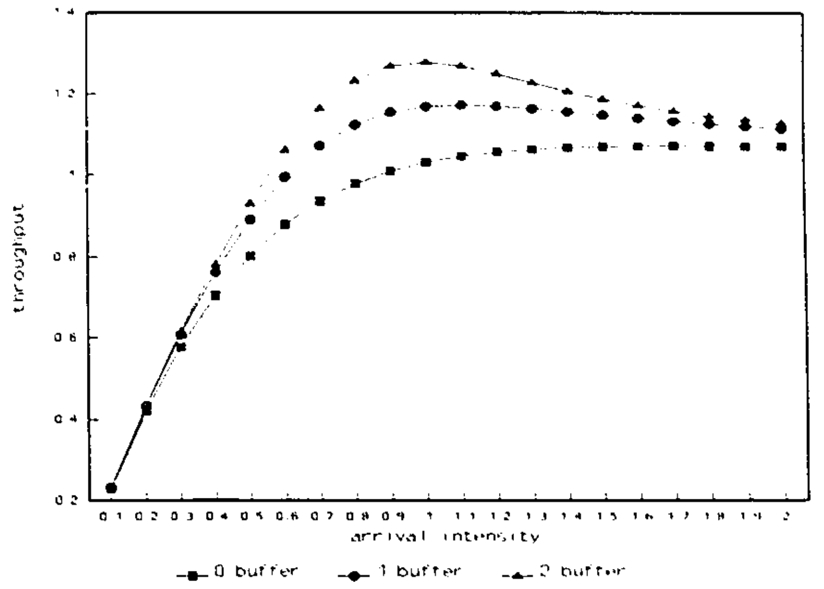


Figure 5.7e Throughputs of downstream machine 1 for *part type 3* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

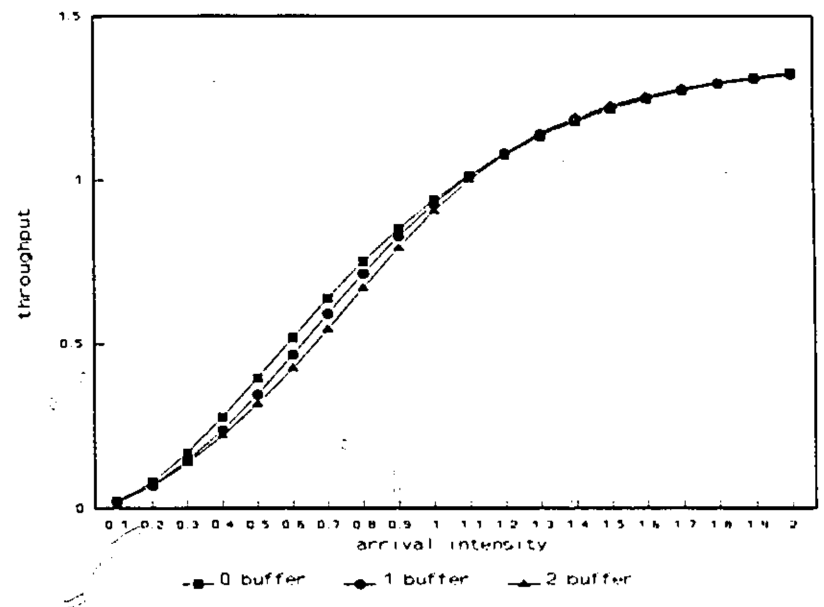


Figure 5.7f Throughputs of downstream machine 2 for *part type 3* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

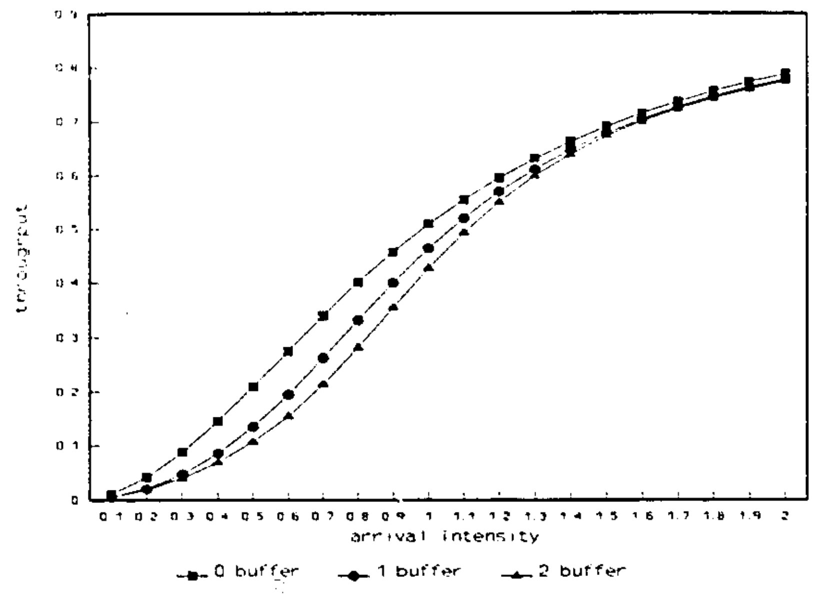


Figure 5.7g Throughputs of downstream machine 1 for *part type 4* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

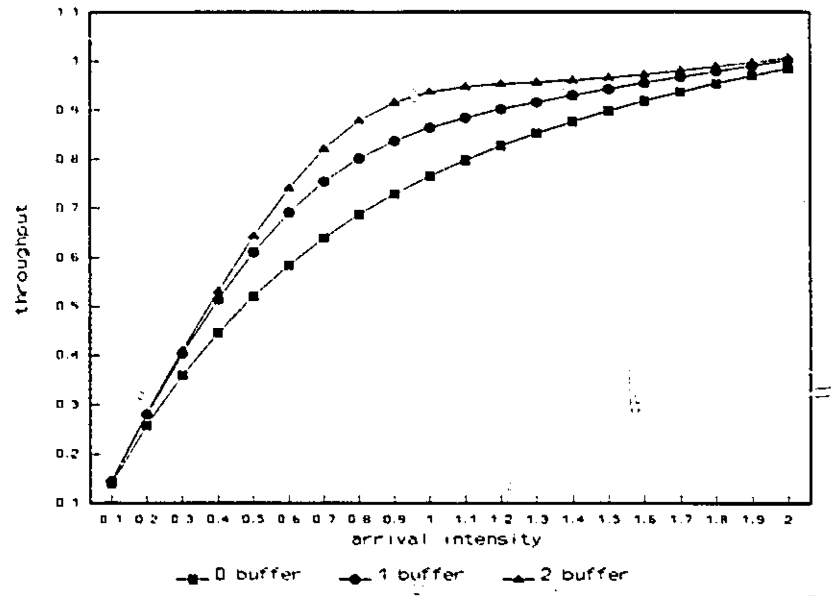


Figure 5.7h Throughputs of downstream machine 2 for *part type 4* when m=4,n=2, and K_i=0,1,2, ∀i∈[1,n].

5.3.3 Influence of Number of Part Types

To see the influence of the "number of part types" on the system performance, it is assumed that the instage buffer size is 0 and the mean processing rates of two downstream machines remain unchanged; the number of part types (number of upstream machines) varies from 1 to 4. The associated mean processing rates of upstream machines are given in **Table 5.3**. PN model III is used to obtain the performance results, as shown in **Figures 5.8a-c.**

As one might expect, with more part types introduced into the system, the system throughput (including all downstream machines for all part types) is increased because the mean processing rates of downstream machines with respect to the newly introduced part types keep increasing. The throughput with respect to each part type is decreased as each downstream machine has to spend less time on a particular part type with more part types in the system. More jobs blocked in the upstream stage is a result of more part types in the system while the number of downstream machines remains the same.

Table 5.3a Mean Processing Rates of Upstream Machines

	1 x 2 cell	2 x 2 cell	3 x 2 cell	4 x 2 cell
Part type 1				0.45 - 9.0
Part type 2			0.47 - 9.4	0.35 - 7.0
Part type 3		0.5 - 10.0	0.33 - 6.6	0.25 - 5.0
Part type 4	0.6 - 12.0	0.3 - 6.0	0.2 - 4.0	0.15 - 3.0

Table 5.3b Mean Processing Rates of Downstream Machines

Part Type	Mean Processing Rate	
1	Machine 1	10.0
	Machine 2	8.0
2	Machine 1	5.0
	Machine 2	9.0
3	Machine 1	6.0
	Machine 2	4.0
4	Machine 1	2.0
	Machine 2	4.0

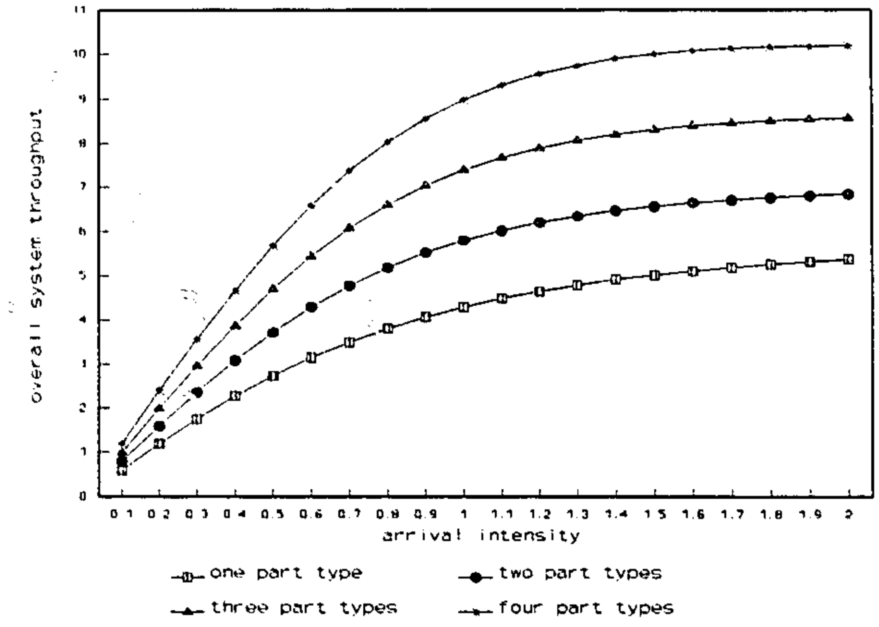


Figure 5.8a Overall throughputs of downstream machines

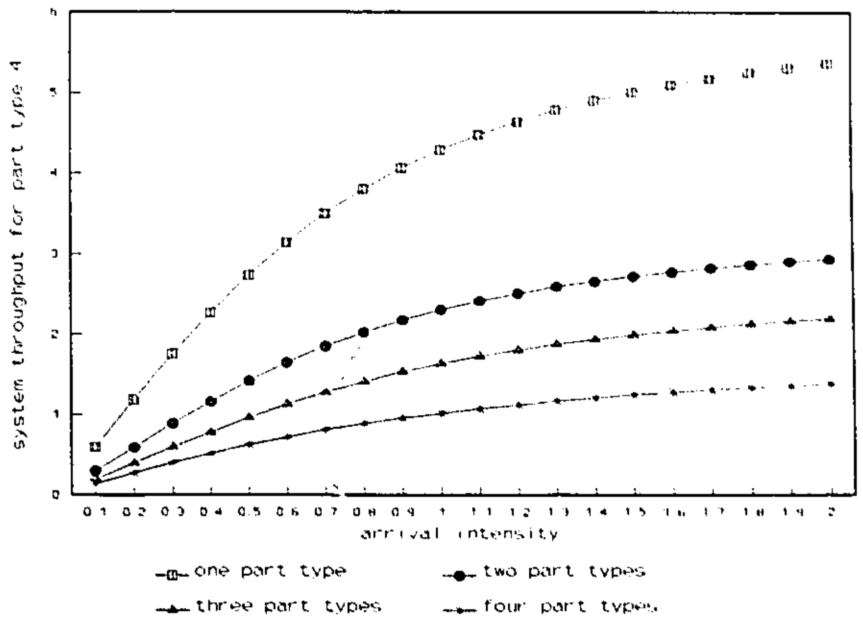


Figure 5.8b Throughputs of downstream machines with respect to *part type 4.*

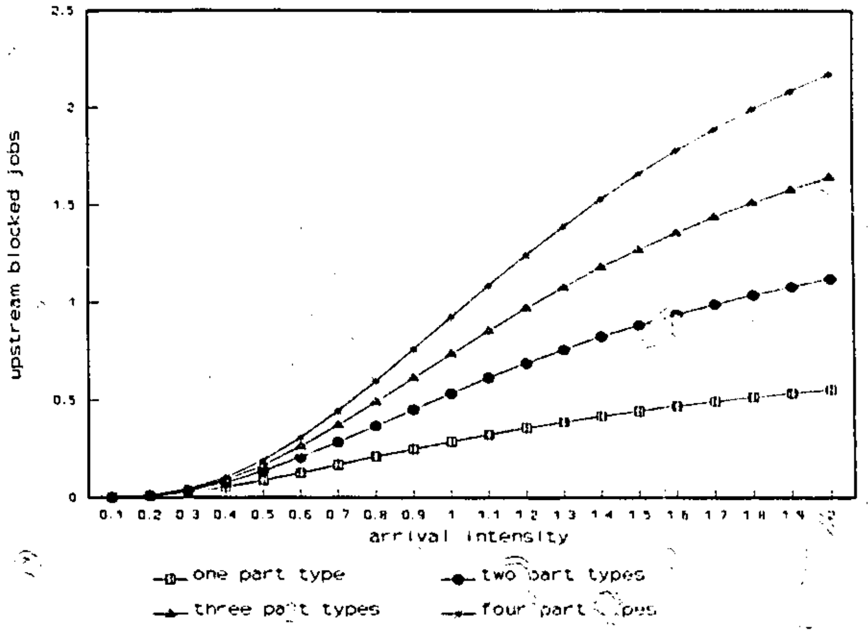


Figure 5.8c Number of blocked jobs in upstream machines

Chapter 6

CONCLUSIONS AND SUGGESTIONS

6.1 Conclusions

This thesis is the first effort to date in investigating the behaviour of an auction-based manufacturing cell with multiple part types and multiple machining centres. A review on the heterarchical control architectures is given in chapter 2.

Instead of the traditional approaches for performance analysis, *i.e.*, analytical models and simulation models, the Generalized Stochastic Petri Nets are applied to model a two-stage FMC with a heterarchical control mechanism. Background knowledge on GSPN is provided in Chapter 3.

Three Petri Net models have been developed in chapter 4:

Model I: single part type n machine with instage buffers,

Model II: m part type n machine with instage buffers, and

Model III: m part type n machine without instage buffers.

Performance measures such as machine throughput and utilization, number of blocked jobs in the upstream machines, and queue lengths in the instage buffers have been defined in terms of GSPN and can be obtained by solving the models using the SPNP package.

Two examples are given in chapter 5 to demonstrate how performance evaluation can be carried out by the PN models developed. Reasonable Results have been obtained, which indicate that the models are working properly. Also, some insight into the behaviour of the auction-based manufacturing cell has been obtained based on the numerical results.

6.2 Suggestions for Future Research

While the numerical results obtained from the PN models yield valuable information concerning the auction behaviour of a two-stage FMC, other aspects of system performance need to be tested, too.

Usually, a part has to go through several stages before it leaves a manufacturing system. The performance of a multi-stage auction-based FMC should be studied.

The models developed in this thesis are for an FMC with "pull" mechanism, *i.e.*, a part is available whenever an upstream machine is free. The performance evaluation of an auction-based system with "push" mechanism is recommended for future research.

Sometimes, a machining centre cannot work on all the part types in a system, but only some of them. The system performance in this case may be investigated.

The optimality of a heterarchically controlled manufacturing system is a research topic suggested by many researchers (Shaw 1985, Duffie *et al.* 1987). This thesis provides a way to obtain the performance of an auction-based FMC, which can be compared with that of centralized control systems, to see how far the local optimization is from its global solution.

In this study, the Earliest Finishing Time rule is applied by parts to select the next machine for its operation. Other heuristics rules such as random selection, fewest operations remaining, shortest processing time (Ravi *et al.* 1991), can be modelled, as well. Comparison of system performance under different rules is another interesting area which deserves further attention.

Exponential time is the major assumption in this research. Other distributions could be accommodated with the Extended Stochastic Petri Nets Package, which is currently under development at Duke University and is expected to be available soon.

To facilitate users, who are not familiar with the SPNP, in analyzing such an auction-based FMC, a software for generating the SPNP code needs to be developed.

REFERENCES

AJMONE MARSAN, M., Balbo, G., Chiola, G., and Conte, G., Applicability of Stochastic Petri Nets to Performance Modelling, Iazeolla, G., Courtois, P.J., and Boxma, O.J. (Editors), *Computer Performance and Reliability*, North-Holland, pp.517-534, 1988.

AJMONE MARSAN, M., Balbo, G., and Conte, G. *Performance Models of Multiprocessor System*, The MIT Press, Chapter 4, 1986.

ANSTISS, P., The implementation and Control of Advanced Manufacturing Systems, *Control and Programming in Advanced Manufacturing,* IFS Publications Ltd., UK, Springer-Verlag, Berlin, 1988.

BOOTH, G.M., *The Distributed System Environment,* McGraw-Hill Book Company, NY 1981.

CIARDO, G. and Muppala,J.K., *Manual for the SPNP Package, Version 3.1,* Duke University, 1992.

CUTOSKY, M.R., Fussell, P.S., and Milligan, Jr., The Design of a Flexible Machining Cell for Small Batch Production, *Journal of Manufacturing Systems*, Vol.3, No.1, pp.39-58, 1984.

DAVIS, R., and Smith, R.G., Negotiation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence,* 20, pp.60-109, 1983.

DESROCHERS, A., *Modelling and Control of Automated Manufacturing Systems*, chapter 5, IEEE Computer Society Press Tutorial, IEEE Computer Society Press, 1990.

DILTS, D.M., Boyd, N.P., and Whorms, H.H., The Evolution of Control Architectures for Automated Control Systems, *Journal of Manufacturing Systems,* 10(1), pp.79-63, 1991.

DUFFIE, N.A., Synthesis of Heterarchical Manufacturing Systems, *Computers in Industry*, Vol.14, pp.167-174, 1990.

DUFFIE, N.A., Chitturi, R., and Mou, J., Fault-Tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities, *Journal of Manufacturing Systems*, Vol. 7, No.4, pp. 315-327, 1988.

DUFFIE, N.A. and Piper, R.S., Non-Hierarchical Control of a Flexible Manufacturing Cell, *Robotics & Computer-Integrated Manufacturing,* Vol.3, No.2, 1987, pp.175-179.

DUFFIE, N.A., An Approach to the Design of Distributed Manufacturing Control Systems, IEEE Transactions on Industry Applications, Vol. IA-18, No.4, pp.435-442, July/August 1982.

FUKUDA, T., Takeda, S., and Hayashi, M., Distributed Expert Systems for Production Control, *Proceedings 1986 IIE Conference*, Dallas, TX, pp.222-228, May 1986.

HATVANY, J., Intelligence and Cooperation in Heterarchic Manufacturing Systems, *Robotics & Computer-Integrated Manufacturing*, Vol.2, No.2, pp.101-104, 1985.

HATVANY, J. and Nemes, L., Intelligent Manufacturing Systems - A Tentative Forecast, In *A Link Between Science and Applications of Automatic Control*, Vol. 2, Niemi, A., Wahlstrom, B., Virkkunen, J. (Eds.), Oxford, Pergamon, pp.895-899, 1978.

HWANG, K. and Briggs, F.A., *Computer Architecture and Parallel processing*, McGraw-Hill, 1984.

JUNGNITZ, H. and Desrochers, A., An Overview of Analytical Performance Analysis Models for Manufacturing Systems, *PED-Vol.53, Design, Analysis, and Control of Manufacturing Cells*, Edited by Black, J.T., Jiang, B.C., and Wiens, G.J., pp. 71-82, ASME 1991.

KNAPP, G. M. and Wang, H.-P.(Ben), Modelling of Automated Storage/Retrieval Systems Using Petri Nets, *Journal of Manufacturing Systems*, Vol. 11, No. 1, pp.20-29, 1992.

KOCHIKAR, V.P. and Narendran, T.T., Modelling Automated Manufacturing Systems Using A Modification of Coloured Petri Nets, *Robotics & Computer-Integrated Manufacturing*, Vol.9, No.3, pp.181-189, 1992.

LEWIS, W., Barash, M.M., and Solberg, J.J., Computer Integrated Manufacturing System Control: A Data Flow Approach, *Journal of Manufacturing Systems*, 6, pp.177-191, 1987.

LEWIS, W., Barash, M.M., and Solberg, J.J., Single Queue Management of a Job Shop as Implemented by a Data Flow Architecture, *Proceedings of the 23rd International Machine Tool Design and Research Conference*, Manchester, pp.415-420, September 1982.

LIN, G.Y. and Solberg, J.J., Integrated Shop Floor Control Using Autonomous Agents, *IIE Transactions*, Vol.24, No.3, pp.57-83, July 1992.

MCGILLEM, C.D., Solberg, J.J., Tanchoco, J.M.A., Midha, A., and Moodie, C.L., Towards an Intelligent Factory Transport System, Invited Presentation at the ORSA/TIMS Meeting, Miani, FL, 27-29 October, 1986.

MOLLOY, M.K., Performance Analysis Using Stochastic Petri Nets, *IEEE Transactions on Computers*, Vol. C-31, No.9, pp.913-917, September, 1982.

PARUNAK, H., Manufacturing Experience with the Contract Net, *Distributed Artificial Intelligence*, Huhns, Michael M. ed., Pitman, London, pp.285-310, 1987.

PARUNAK, H., VanDyke, B.W., Irish, J.K., and Lozo, P.W., Fractal Actors for Distributed Manufacturing Control, *Proceedings of the IEEE Second Conference on AI Applications,* Miami Beach, FL, pp.653-660, December 1985.

PEPERSON, J.L., *Petri Net Theory and the Modelling of Systems,* Prentice-Hall, Inc., Englewood Cliffs, 1981.

PIMENTAL, J.R. *Communication Networks for Manufacturing*, Prentice Hall, Englewood Cliffs, NJ, 1990.

RANA, S.P. and Taneja, S.K., A Distributed Architecture for Automated Manufacturing Systems, *International Journal of Advanced Manufacturing Technology*, Vol.3, No.5, pp. 81-98, 1988.

RAVI, T., Lashkari, R.S., and Dutta, S.P., Selection of Scheduling Rules in FMSs - A Simulation Approach, *International Journal of Advanced Manufacturing Technology*, No.6, pp.246-262, 1991.

SHAW, M.J., Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Networked Decision Making, *Journal of Manufacturing Systems,* Vol.7, No.2, pp.83-94, 1988.

SHAW, M.J., Task Bidding and Distributed Planning in Flexible Manufacturing, *Proceedings of the Second Conference on Artificial Intelligence Applications,* pp.184-189, IEEE, Miami Beach, FL, 1985.

SHAW, M.J. and Whinston, A.B., Automatic Planning and Flexible Scheduling: A Knowledge Base, *IEEE International Conference on Robotics and Automation,* St Louis, Missouri, pp.890-894, 1985.

SMITH, R.G. and Davis, R., Frameworks for Cooperation in Distributed Problem Solving, *IEEE International Conference on Robotics and Automation,* St Louis, Missouri, pp.890-894, 1985.

SMITH, R.G., The contract net protocol: high-level communication and control in a distributed problem solver, *IEEE Trans. Computers, C-29:*1104-1113, 1980.

TENG, S.-H., Cellular Manufacturing Systems Modelling: The Petri Net Approach, *Journal of Manufacturing Systems*, Vol.9, No.1, pp.45-54, 1990.

TIRPAK, T. M., Deligiannis, S.J., and Davis, W.J., Real-Time Scheduling in Flexible Manufacturing, *Manufacturing Review*, vol.5, no.3, pp.193-212, September 1992.

UPTON, D.M., Barash, M.M., and Matheson, A.M., Architectures and Auctions in Manufacturing, *International Journal of Computer Integrated Manufacturing*, 1991, Vol.4, No.1, 23-33, 1991.

UPTON, D.M., A Flexible Structure for Computer-Controlled Manufacturing Systems, *Manufacturing Review*, Vol.5, No.1, pp.58-72, March 1992.

VISWANADHAM, N. and Narahari, Y., *Performance Modelling of Automated Manufacturing Systems*, Prentice Hall, NJ, 1992.

APPENDIX - A

SPNP PROGRAMMES

A.1 SPNP Programm for Model I & II: Buffer Size = 1

```
/*----------------------------------------------------------------------------------------------
 Features: heterarchical control of a FMC considering effect of buffer size (=1)
    •  Input data from file "data_in_1buf";
    •  m: # of upstream machines;
    •  n: # of downstream machines;
    •  rate_tp1(i), i=1,...,m: processing rate of upstream machine i;
    •  rate_tp2(i,j), i=1,...,n, j=1,...,m: processing rate of downstream machine i for
       part type j;
    •  initial markings = 1 for pf1(j) and pf2(i), i=1,...,n, j=1,...,m.
----------------------------------------------------------------------------------------------*/
#include "user.h"
#include <math.h>
#define min(x,y) ((x<y)? x: y)

int m,n,bf,s,q;
float rate_tp1[10], rate_tp2[10][10], lamda, lamda_n, lamda_d;

parameters()
{
  FILE *fp;
  int i, j;

  iopt(IOP_METHOD, VAL_GASEI);
  iopt(IOP_PR_FULL_MARK, VAL_YES);
  iopt(IOP_PR_MARK_ORDER, VAL_CANONIC);
  iopt(IOP_PR_MC_ORDER, VAL_TOFROM);
  iopt(IOP_PR_MC, VAL_NO);
  iopt(IOP_PR_PROB, VAL_NO);
  iopt(IOP_MC, VAL_CTMC);
  iopt(IOP_PR_RSET, VAL_YES);
  iopt(IOP_PR_RGRAPH, VAL_YES);
  iopt(IOP_ITERATIONS, 2000);
  fopt(FOP_PRECISION, 1.0e-3);
  fopt(FOP_ABS_RET_M0, 0.0);

  /* input data from file "data_in_1buf" */
  fp = fopen("data_in_1buf", "r");
  fscanf(fp, "%d", &m);
  fscanf(fp, "%d", &n);
  for(j=0; j<m; j++) {
    fscanf(fp, "%f", &rate_tp1[j]);
  }

  for(i=0; i<n; i++)
  for(j=0; j<m; j++) {
    fscanf(fp, "%f", &rate_tp2[i][j]);
  }
  fclose(fp);

  /* lamda = arrival intensity */
  lamda=0.0;
  lamda_n=0.0;
  lamda_d=0.0;
  for(j=0; j<m; j++)
    lamda_n += rate_tp1[j];

  for(i=0; i<n; i++)
  for(j=0; j<m; j++)
    lamda_d += rate_tp2[i][j];

  lamda = m * lamda_n / lamda_d;
}

/* random switches */
probability_type prob_tbf2(i,j)
{
  float p;
  p=mark_1("pi1", j) * mark_1("pbf", i) /enabled_2("tbf2", SUM, SUM);
  if(p < 1.0e-10)
    return 1.0e-10;
  else
    return(p);
}

probability_type prob_tm2(i, j)
{
  float p;
  p = mark_2("pbf2", i, j) * mark_1("pf2", i) /enabled_2("tm2", SUM, SUM);
  if(p < 1.0e-10)
    return 1.0e-10;
  else
    return(p);
}

/* EFT rule */
enabling_type enb_tbf2(i, j)
{
  float PT1, PT;
  int  k, h, K;

  if(mark_1("pi1", j) == 0) return(0);
  else if (mark_1("pbf",i) == 0) return(0);
  else {
    PT1=1.0e+10;
    for(k=0; k<n; k++) {
      if( mark_1("pbf", k) != 0 ) {
        PT = 1.0 / rate_tp2[k][j];
        for(h=0; h<m; h++) {
          PT += (mark_2("pbf2", k, h) + mark_2("pb2", k, h)) /
                rate_tp2[k][h];
        }
        if(PT<PT1) {
          PT1 = PT;
          K = k;
        }
      }
    }
    if (K==i) return(1);
    else return(0);
  }
}

net()
{
  int i,j;

  /* places and transitions */
  /* upstream */
  place_1("pf1", m);
  place_1("pi1", m);
  trans_1("tp1", m);

  /* downstream */
  place_1("pbf", n);
  place_1("pf2", n);
  place_2("pbf2", n, m);
  place_2("pb2", n, m);
  trans_2("tbf2", n, m);
  priority_2("tbf2", ALL, ALL, 1);
  trans_2("tm2", n, m);
  priority_2("tm2", ALL, ALL, 2);
  trans_2("tp2", n, m);

   /* initial markings */
```

```
init_1("pf1", ALL, 1);
init_1("pbf", ALL, 1);
init_1("pf2", ALL, 1);

/* arcs */
/* upstream */
for(i=0; i<m; i++) {
  iarc_1_1("tp1",i, "pf1", i);
  oarc_1_1("tp1",i, "pi1", i);
}

/* downstream */
for(i=0; i<n; i++)
for(j=0; j<m; j++) {
  iarc_2_1("tbf2", i, j, "pi1", j);
  iarc_2_1("tbf2", i, j, "pbf", i);
  oarc_2_1("tbf2", i, j, "pf1", j);
  oarc_2_2("tbf2", i, j, "pbf2", i, j);

  iarc_2_1("tm2", i, j, "pf2", i);
  iarc_2_2("tm2", i, j, "pbf2", i, j);
  oarc_2_1("tm2", i, j, "pbf", i);
  oarc_2_2("tm2", i, j, "pb2", i, j);

  iarc_2_2("tp2", i, j, "pb2", i, j);
  oarc_2_1("tp2", i, j, "pf2", i);
}

/* rates for timed transitions */
for(j=0; j<m; j++)
  rateval_1("tp1", j, rate_tp1[j]);

for(i=0; i<n; i++)
for(j=0; j<m; j++)
  rateval_2("tp2", i, j, rate_tp2[i][j]);


/* marking dependent firing probabilities
      for immediate transitions */
for(i=0; i<n; i++)
for(j=0; j<m; j++) {
  probfun_2("tbf2", i, j, prob_tbf2 );
  probfun_2("tm2", i, j, prob_tm2 );
}

/* marking dependent enabling functions */
for(i=0; i<n; i++)
```

```
  for(j=0; j<m; j++)
    enabling_2("tbf2", i, j, enb_tbf2);
}

assert() {
  return(RES_NOERR);
}

ac_init() {
  fprintf(stderr, "\nPetri nets for FMC\n\n");
  pr_net_info();
}

ac_reach() {
  fprintf(stderr, "\nThe reachability graph has been generated\n\n");
  pr_rg_info();
}

reward_type qlength_u() { return(mark_1("pi1",q));}
reward_type util_u() { return(enabled_1("tp1",q));}
reward_type tput_u() { return(rate_1("tp1",q));}

reward_type qlength_d() {
  int  q_d;
 { q_d  = mark_2("pbf2",s,SUM);}
  return(q_d);
}

reward_type util_d() {
  float u_d;
  { u_d  = enabled_2("tp2", s,SUM);}
  return(u_d);
}

reward_type tput_d() {
  float t_d;
  { t_d  = rate_2("tp2", s,SUM);}
  return(t_d);
}

reward_type t_put2_d() {return (rate_2("tp2",s,q));}

ac_final() {
  float ql_u,ut_u,tp_u, ql_d, ut_d, tp_d, tp2_d;
  FILE *fp1, *fp2;

  pr_mc_info();
```

```
pr_std_average();

/* output data to file "data_out_1buf_u/d" */
fp1 = fopen("data_out_1buf_u", "w");
fprintf(fp1, "\n");
fprintf(fp1, "%f  ", lamda);
for(q=0; q<m; q++) {
  ql_u=expected(qlength_u);
  fprintf(fp1,"%f  ",ql_u);
}
for(q=0; q<m; q++) {
   ut_u=expected(util_u);
   fprintf(fp1,"%f  ",ut_u);
}
for(q=0; q<m; q++) {
   tp_u=expected(tput_u);
   fprintf(fp1,"%f  ",tp_u);
}

fclose(fp1);

fp2 = fopen("data_out_1buf_d", "w");
fprintf(fp2, "\n");
fprintf(fp2, "%f  ", lamda);
for(s=0; s<n; s++) {
  ql_d=expected(qlength_d);
  fprintf(fp2,"%f  ",ql_d);
}
for(s=0; s<n; s++) {
   ut_d=expected(util_d);
   fprintf(fp2,"%f  ",ut_d);
}
for(s=0; s<n; s++) {
   tp_d=expected(tput_d);
   fprintf(fp2,"%f  ",tp_d);
}
for(s=0; s<n; s++)
for(q=0; q<m; q++) {
  tp2_d=expected(t_put2_d);
  fprintf(fp2,"%f  ", tp2_d);
}

fclose(fp2);
}
```

A.2 SPNP Programm for Model I & II: Buffer Size = 2

```
/*----------------------------------------------------------------------------------------------------
 Feature: heterarchical control of a FMC considering effect of buffer size(=2)
    •  Input data from file "data_in_2buf"; Output data to "data_out_2buf";
    •  m: # of upstream machines;
    •  n: # of downstream machines;
    •  bf: buffer size for Pbf(i), i=1,2,...n;
    •  rate_tp1(i), i=1,...,m: processing rate of upstream machine i;
    •  rate_tp2(i,j), i=1,...,n, j=1,...,m: processing rate of downstream machine i for
       part type j;
    •  initial markings = 1 for pf1(j) and pf2(i), i=1,...,n, j=1,...,m.
----------------------------------------------------------------------------------------------------*/
#include "user.h"
#include <math.h>
#define min(x,y) ((x<y)? x: y)

int m,n,bf,s,q;
float rate_tp1[10], rate_tp2[10][10], lamda, lamda_n, lamda_d;

parameters()
{
  FILE *fp;
  int i, j;

  iopt(IOP_METHOD, VAL_GASEI);
  iopt(IOP_PR_FULL_MARK, VAL_YES);
  iopt(IOP_PR_MARK_ORDER, VAL_CANONIC);
  iopt(IOP_PR_MC_ORDER, VAL_TOFROM);
  iopt(IOP_PR_MC, VAL_NO);
  iopt(IOP_PR_PROB, VAL_NO);
  iopt(IOP_MC, VAL_CTMC);
  iopt(IOP_PR_RSET, VAL_NO);
  iopt(IOP_PR_RGRAPH, VAL_NO);
  iopt(IOP_ITERATIONS, 2000);
  fopt(FOP_PRECISION, 1.0e-3);
  fopt(FOP_ABS_RET_M0, 0.0);

  /* input data from file "data_in_2buf" */
  fp = fopen("data_in_2buf", "r");
  fscanf(fp, "%d", &m);
  fscanf(fp, "%d", &n);
  fscanf(fp, "%d", &bf);
  for(j=0; j<m; j++) {
    fscanf(fp, "%f", &rate_tp1[j]);
  }
 for(i=0; i<n; i++)
 for(j=0; j<m; j++) {
   fscanf(fp, "%f", &rate_tp2[i][j]);
 }
 fclose(fp);

 /* lamda = arrival intensity */
 lamda=0.0;
 lamda_n=0.0;
 lamda_d=0.0;
 for(j=0; j<m; j++)
   lamda_n += rate_tp1[j];

 for(i=0; i<n; i++)
 for(j=0; j<m; j++)
   lamda_d += rate_tp2[i][j];

 lamda = m * lamda_n / lamda_d;
}

/* random switches */
probability_type prob_tbf2(i,j)
{
  float p;
  p=mark_1("pi1", j) * mark_2("pbf", i, bf-1) /enabled_2("tbf2", SUM, SUM);
  if(p < 1.0e-10)
    return 1.0e-10;
  else
    return(p);
}

probability_type prob_t1(i,j)
{
  float p;
  p=mark_2("p2", i, j) * mark_2("pbf", i, 0) /enabled_2("t1", SUM, SUM);
  if(p < 1.0e-10)
    return 1.0e-10;
  else
    return(p);
}


probability_type prob_tm2(i, j)
{
  float p;
   p = mark_2("p1", i, j ) * mark_1("pf2", i) /enabled_2("tm2", SUM, SUM);
   if(p < 1.0e-10)
```

```
    return 1.0e-10;
  else
    return(p);
}

/* EFT rule */
enabling_type enb_tbf2(i, j)
{
  float PT1, PT;
  int k, h, K;

  if(mark_1("pi1", j) == 0) return(0);
  else if (mark_2("pbf",i,bf-1) == 0) return(0);
  else {
    PT1=1.0e+10;
    for(k=0; k<n; k++) {
      if( mark_2("pbf", k, bf-1) != 0 ) {
        PT = 1.0 / rate_tp2[k][j];   /*self processing time*/
        for(h=0; h<m; h++) {
          PT += ( mark_2("p2", k, h) + mark_2("p1", k, h) +
                mark_2("pb2", k, h) ) / rate_tp2[k][h];
        }
        if(PT<PT1) {
          PT1 = PT;
          K = k;
        }
      }
    }
    if (K==i) return(1);
    else return(0);
  }
}

net()
{
  int i,j ;

  /* places and transitions */
  /* upstream */
  place_1("pf1", m);
  place_1("pi1", m);
  trans_1("tp1", m);

  /* downstream */
  place_2("pbf", n, bf);
  place_2("p2", n, m );
  place_2("p1", n, m);
```

```
place_1("pf2", n);
place_2("pb2", n, m);

trans_2("tbf2", n, m);
priority_2("tbf2", ALL, ALL, 1);
trans_2("t1", n, m);
priority_2("t1", ALL, ALL, 2);
trans_2("tm2", n, m);
priority_2("tm2", ALL, ALL, bf+1);
trans_2("tp2", n, m);


/* initial markings */
init_1("pf1", ALL, 1);
init_2("pbf", ALL, ALL, 1);
init_1("pf2", ALL, 1);

/* arcs */
/* upstream */
for(i=0; i<m; i++) {
  iarc_1_1("tp1",i, "pf1", i);
  oarc_1_1("tp1",i, "pi1", i);
}

/* downstream */
for(i=0; i<n; i++)
for(j=0; j<m; j++) {
  iarc_2_1("tbf2", i, j, "pi1", j);
  iarc_2_2("tbf2", i, j, "pbf", i, bf-1);
  oarc_2_2("tbf2", i, j, "p2", i, j);
  oarc_2_1("tbf2", i, j, "pf1", j);

  iarc_2_2("t1", i,j , "p2", i,j );
  iarc_2_2("t1", i,j , "pbf", i,0 );
  oarc_2_2("t1", i,j , "p1", i,j );
  oarc_2_2("t1", i,j , "pbf", i, 1);

  iarc_2_2("tm2", i, j, "p1", i, j);
  iarc_2_1("tm2", i, j, "pf2", i);
  oarc_2_2("tm2", i, j, "pb2", i, j);
  oarc_2_2("tm2", i, j, "pbf", i, 0);

  iarc_2_2("tp2", i, j, "pb2", i, j);
  oarc_2_1("tp2", i, j, "pf2", i);
}

/* rates for timed transitions */
```

```
  for(j=0; j<m; j++)
    rateval_1("tp1", j, rate_tp1[j]);

  for(i=0; i<n; i++)
  for(j=0; j<m; j++)
    rateval_2("tp2", i, j, rate_tp2[i][j]);

  /* marking dependent firing probabilities
        for immediate transitions */
  for(i=0; i<n; i++)
  for(j=0; j<m; j++) {
    probfun_2("tbf2", i, j, prob_tbf2 );
    probfun_2("t1", i,j, prob_t1);
    probfun_2("tm2", i, j, prob_tm2 );
   }

  /* marking dependent enabling functions */
  for(i=0; i<n; i++)
  for(j=0; j<m; j++) {
    enabling_2("tbf2", i, j, enb_tbf2);
  }
}

assert() {
  return(RES_NOERR);
}

ac_init() {
  fprintf(stderr, "\nPetri nets for FMC\n\n");
  pr_net_info();
}

ac_reach() {
  fprintf(stderr, "\nThe reachability graph has been generated\n\n");
  pr_rg_info();
}

reward_type qlength_u() { return(mark_1("pi1",q));}
reward_type util_u() { return(enabled_1("tp1",q));}
reward_type tput_u() { return(rate_1("tp1",q));}

reward_type qlength_d() {
  int  q_d;
 { q_d  = mark_2("p2",s,SUM) + mark_2("p1", s, SUM);}
  return(q_d);
}
```

```
reward_type util_d() {
  float u_d;
  { u_d  = enabled_2("tp2", s.SUM);}
  return(u_d);
}

reward_type tput_d() {
  float t_d;
  { t_d  = rate_2("tp2", s.SUM);}
  return(t_d);
}

reward_type t_put2_d() {return (rate_2("tp2",s,q));}

ac_final() {
  float ql_u,ut_u,tp_u, ql_d, ut_d, tp_d, tp2_d;
  FILE *fp1, *fp2;

  pr_mc_info();
  pr_std_average();

  /* output data to file "data_out_2buf_u/d" */

  fp1 = fopen("data_out_2buf_u", "w");
  fprintf(fp1, "\n");
  fprintf(fp1, "%f  ", lamda);
  for(q=0; q<m; q++) {
    ql_u=expected(qlength_u);
    fprintf(fp1,"%f  ",ql_u);
  }
  for(q=0; q<m; q++) {
     ut_u=expected(util_u);
     fprintf(fp1,"%f  ",ut_u);
  }
  for(q=0; q<m; q++) {
     tp_u=expected(tput_u);
     fprintf(fp1,"%f  ",tp_u);
  }
  fclose(fp1);

  fp2 = fopen("data_out_2buf_d", "w");
  fprintf(fp2, "\n");
  fprintf(fp2, "%f  ", lamda);
  for(q=0; q<n; q++) {
    ql_d=expected(qlength_d);
    fprintf(fp2,"%f  ",ql_d);
  }
```

```
   for(q=0; s<n; q++) {
      ut_d=expected(util_d);
      fprintf(fp2,"%f  ",ut_d);
   }
   for(q=0; s<n; q++) {
      tp_d=expected(tput_d);
      fprintf(fp2,"%f  ",tp_d);
   }
   for(s=0; s<n; s++)
   for(q=0; q<m; q++) {
     tp2_d=expected(t_put2_d);
     fprintf(fp2,"%f  ", tp2_d);
   }
   fclose(fp2);
 }
```

A.3 SPNP Programm for Model III: No Instage Buffers

```
/*-------------------------------------------------------------------------------------------------*/
Feature: heterarchical control of a FMC considering the case of no instage buffers
        •   Input data from file "data_in_0buf";
        •   m: # of upstream machines;
        •   n: # of downstream machines;
        •   rate_tp1(i), i=1,...,m: processing rate of upstream machine i;
        •   rate_tp2(i,j), i=1,...,n, j=1,...,m: processing rate of downstream machine
            i for part type j;
        •   initial markings = 1 for pf1(j) and pf2(i), i=1,...,n, j=1,...,m.
/*-------------------------------------------------------------------------------------------------*/
#include "user.h"
#include <math.h>
#define min(x,y) ((x<y)? x: y)

int m,n,bf,q,s;
float rate_tp1[10], rate_tp2[10][10], lamda, lamda_n, lamda_d;

parameters()
{
  FILE *fp;
  int i, j;

  iopt(IOP_METHOD, VAL_GASEI);
  iopt(IOP_PR_PROB, VAL_NO);
  iopt(IOP_MC, VAL_CTMC);
  iopt(IOP_ITERATIONS, 2000);
  fopt(FOP_PRECISION, 1.0e-10);
  fopt(FOP_ABS_RET_M0, 0.0);

  /* input data from file "data_in_0buf" */
  fp = fopen("data_in_0buf", "r");
  fscanf(fp, "%d", &m);
  fscanf(fp, "%d", &n);
  for(j=0; j<m; j++) {
    fscanf(fp, "%f", &rate_tp1[j]);
    printf("%d : %f\n", j, rate_tp1[j]);
  }
  for(i=0; i<n; i++)
    for(j=0; j<m; j++) {
    fscanf(fp, "%f", &rate_tp2[i][j]);
    printf("%d %d: %f\n", i, j, rate_tp2[i][j]);
    }
  fclose(fp);
  /* lamda = arrival intensity */
  lamda=0.0;
  lamda_n=0.0;
  lamda_d=0.0;
  for(j=0; j<m; j++)
    lamda_n += rate_tp1[j];

  for(i=0; i<n; i++)
  for(j=0; j<m; j++)
    lamda_d += rate_tp2[i][j];

  lamda = m * lamda_n / lamda_d;
}

/* EFT rule */
enabling_type enb_tm2(i, j)
{
  float PT1, PT;
  int  k, h, K;

  if(mark_1("pi1", j) == 0) return(0);
  else if (mark_1("pf2",i) == 0) return(0);
  else {
    PT1=1.0e+10;
    for(k=0; k<n; k++) {
      if ( mark_1("pf2",k) != 0) {
        PT =  1.0 / rate_tp2[k][j];
        for(h=0; h<m; h++) {
          PT +=  mark_2("pb2", k, h) / rate_tp2[k][h];
        }
        if(PT<PT1) {
        PT1 = PT;
        K = k;
        }
      }
    }
    if (K==i) return(1);
    else return(0);
  }
}

/* random switch */
/* firing probability of transition tm2(i,j)*/
probability_type prob_tm2(i, j)
{
  float p;
  p = 1.0 /enabled_2("tm2", SUM, SUM);
  return(p);
}

net()
{
  int i,j;

  /* places and transitions */
  /* upstream */
  place_1("pf1", m);
  place_1("pi1", m);
  trans_1("tp1", m);

  /* downstream */
  place_1("pf2", n);
  place_2("pb2", n, m);
  trans_2("tm2", n, m);
  priority_2("tm2", ALL, ALL, 1);
  trans_2("tp2", n, m);

  /* initial markings */
  init_1("pf1", ALL, 1);
  init_1("pf2", ALL, 1);

  /* arcs */
  /* upstream */
  for(i=0; i<m; i++) {
    iarc_1_1("tp1",i, "pf1", i);
    oarc_1_1("tp1",i, "pi1", i);
  }

  /* downstream */
  for(i=0; i<n; i++)
  for(j=0; j<m; j++) {
    iarc_2_1("tm2", i, j, "pi1", j);
    iarc_2_1("tm2", i, j, "pf2", i);
    oarc_2_1("tm2", i, j, "pf1", j);
    oarc_2_2("tm2", i, j, "pb2", i, j);

    iarc_2_2("tp2", i, j, "pb2", i, j);
    oarc_2_1("tp2", i, j, "pf2", i);
  }

  /* rates for timed transitions */
  for(j=0; j<m; j++) {
    rateval_1("tp1", j, rate_tp1[j]);
  }
  for(i=0; i<n; i++)
  for(j=0; j<m; j++){
    rateval_2("tp2", i, j, rate_tp2[i][j]);
  }

  /* marking dependent firing probabilities
              for immediate transitions */
  for(i=0; i<n; i++)
  for(j=0; j<m; j++) {
    probfun_2("tm2", i, j, prob_tm2 );
    enabling_2("tm2", i, j, enb_tm2);
  }
}

assert() {
  return(RES_NOERR);
}

ac_init() {
  fprintf(stderr, "\nPetri nets for FMC\n\n");
  pr_net_info();
}

ac_reach() {
  fprintf(stderr, "\nThe reachability graph has been generated\n\n");
  pr_rg_info();
}

reward_type qlength_u() { return(mark_1("pi1",q));}
reward_type util_u() { return(enabled_1("tp1",q));}
reward_type tput_u() { return(rate_1("tp1",q));}
reward_type util_d() { return(enabled_2("tp2", s, SUM));}
reward_type tput_d() { return(rate_2("tp2", s, SUM));}
reward_type t_put2_d() {return (rate_2("tp2", s,q));}

ac_final() {
  float ql_u,ut_u,tp_u, ut_d, tp_d, tp2_d;
  FILE *fp1, *fp2;

  pr_mc_info();
  pr_std_average();

/* output data to file "data_out_0buf_u/d" */
  fp1 = fopen("data_out_0buf_u", "w");
  fprintf(fp1, "\n");
  fprintf(fp1, "%f  ", lamda);
  for(q=0; q<m; q++) {
   ql_u=expected(qlength_u);
   fprintf(fp1,"%f  ",ql_u);
 }
 for(q=0; q<m; q++) {
    ut_u=expected(util_u);
    fprintf(fp1,"%f  ",ut_u);
 }
 for(q=0; q<m; q++) {
    tp_u=expected(tput_u);
    fprintf(fp1,"%f  ",tp_u);
 }
 fclose(fp1);

 fp2 = fopen("data_out_0buf_d", "w");
 fprintf(fp2, "\n");
 fprintf(fp2, "%f  ", lamda);
 for(s=0; s<n; s++) {
    ut_d=expected(util_d);
    fprintf(fp2,"%f  ",ut_d);
 }
 for(s=0; s<n; s++) {
    tp_d=expected(tput_d);
    fprintf(fp2,"%f  ",tp_d);
 }
 for(s=0; s<n; s++)
 for(q=0; q<m; q++) {
   tp2_d=expected(t_put2_d);
   fprintf(fp2,"%f  ", tp2_d);
 }
 fclose(fp2);
}
```

VITA AUCTORIS

NAME:	Qiong Zhou
PLACE OF BIRTH:	Jiangsu, P.R. China
YEAR OF BIRTH:	1962
EDUCATION:	Jing Gong High School, Beijing, P.R. China 1978 - 1980
	Beijing Institute of Technology, Beijing, P.R. China 1980 - 1984, B.Eng.
	Beijing Institute of Technology, Beijing, P.R. China 1984 - 1987, M.Eng.
	Beijing Institute of System Engineering, Chinese Academy of Space Technology, Beijing, P.R. China 1987 - 1991, Engineer
	University of Windor, Windsor, Ontario, Canada 1991 - 1993, M.A.Sc.