Hierarchical Modeling of Manufacturing Systems Using Max-Plus Algebra

A dissertation presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Doctor of Philosophy

Aleksey A. Imaev
November 2009
© 2009 Aleksey A. Imaev. All Rights Reserved.
This dissertation titled
Hierarchical Modeling of Manufacturing Systems Using Max-Plus Algebra

by
ALEKSEY A. IMAEV

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

______________________________
Robert P. Judd
Professor and Chair, Industrial and Systems Engineering

______________________________
Dennis Irwin
Dean, Russ College of Engineering and Technology
ABSTRACT

IMAEV, ALEKSEY A., Ph.D., November 2009, Electrical Engineering

Hierarchical Modeling of Manufacturing Systems Using Max-Plus Algebra (114 pp.)

Director of Dissertation: Robert P. Judd

The dissertation presents a novel hierarchical block-diagram modeling framework for manufacturing systems. A block can be a single manufacturing operation, a single machine, a single part or a factory. Each block has three inputs and three outputs and is represented by a set of linear max-plus algebraic equations. A complex manufacturing system can be modeled as a network of basic manufacturing blocks. Routing of parts and resources through the block diagram graphically corresponds to machine-flow and resource-flow interconnection of blocks and is mathematically modeled by part-flow and machine-flow interconnection matrices, respectively. A formula for composing a network of manufacturing blocks into a single manufacturing block is derived. The model can be used for: (a) performance evaluation, (b) deadlock detection, (c) structural analysis, (d) scheduling, (e) design, and (f) control of manufacturing systems.

The dissertation develops an elegant analysis tool called a matrix signal flow graph (MSFG) over max-plus algebra (also called a synchronous MSFG) for these models. New topological methods for evaluating gains of synchronous MSFGs are presented. Synchronous MSFG provide a straightforward way to covert the graphical block-diagram representation of the system to the max-plus algebraic view.

The dissertation also shows that in the case of a permutation flow shop, an inverse Monge matrix represents the resulting algebraic equations for the system. The dissertation proves that the class of inverse Monge matrices is closed under max-plus algebraic multiplication, and provides an efficient algorithm for computing an eigenvector of an inverse Monge matrix. These properties allow for efficient computation of performance characteristics of permutation flow shops.
Approved: 

Robert P. Judd
Professor and Chair, Industrial and Systems Engineering
Acknowledgements

This dissertation would not have been possible without help of many people. First of all, I would like to express my gratitude to my dissertation advisor, Dr. Robert P. Judd. Thank you for the expertise, guidance and support provided throughout all stages of this research. Special thanks to all the members of my dissertation committee, Dr. Kenneth Cutright, Dr. William Kaufman, Dr. Douglas Lawrence, Dr. Dušan N. Šormaz, and Dr. Constantinos Vassiliadis. Your assistance, insight and suggestions were essential for the completion of this work. Finally, I sincerely thank my parents and my siblings for their continuing encouragement, love and support.
# Table of Contents

Abstract .................................................................................................................. 3

Acknowledgements .................................................................................................. 5

List of Figures ......................................................................................................... 9

1 Introduction ......................................................................................................... 11

2 Literature Review ............................................................................................... 14
   2.1 Automata Modeling Formalism ...................................................................... 14
   2.2 Timed Petri Nets and Timed Event Graphs ...................................................... 15
   2.3 Queueing Networks ....................................................................................... 17
   2.4 Discrete Event Simulation ............................................................................. 17
   2.5 Max-Plus Algebra .......................................................................................... 18

3 Problem Statement ............................................................................................... 20
   3.1 Limitations of Existing Models ..................................................................... 20
   3.2 Objectives and Contributions ....................................................................... 21
      3.2.1 Hierarchical Model of Manufacturing Systems ...................................... 22
      3.2.2 Topological Methods for Performance Evaluation ................................. 22
      3.2.3 Properties of Inverse Monge Matrices .................................................... 23

4 Max-Plus Algebra ................................................................................................ 26
   4.1 Basics ............................................................................................................ 26
   4.2 Communication Graph ................................................................................... 27
   4.3 Definitions and interpretations of $A^{\otimes k}$, $A^+$, and $A^*$ .................... 28
   4.4 Eigenvectors and Eigenvalues ....................................................................... 29
      4.4.1 Eigenvalue ............................................................................................. 29
      4.4.2 Eigenvectors ......................................................................................... 30

5 The Model ............................................................................................................ 31
   5.1 General Block ................................................................................................ 31
   5.2 Composition of Blocks .................................................................................. 33

6 Basic Manufacturing Blocks .............................................................................. 43
   6.1 Single machine processing single part ............................................................ 43
   6.2 Assembly machine ......................................................................................... 44
   6.3 Disassembly machine .................................................................................... 45
   6.4 Unit capacity buffer ...................................................................................... 46
   6.5 Random access buffer with infinite capacity ............................................... 46
6.6 Single machine processing a part: there is an infinite random access buffer in front of the machine ............................................. 47

7 Numerical Example ............................................................. 48
  7.1 Problem Statement ......................................................... 48
  7.2 Job Shop Model ............................................................. 50

8 Analytical Tools ................................................................. 53
  8.1 Deadlock Detection .......................................................... 53
  8.2 Modeling Approaches ...................................................... 54
  8.3 Cyclic Systems ............................................................... 55
  8.4 Performance Evaluation ..................................................... 57
  8.5 Structural Information ..................................................... 58
  8.6 Application to Scheduling ................................................. 58

9 Line Applications ............................................................... 59
  9.1 One Part and $M$ Resources .............................................. 59
     9.1.1 Each $m_i$ is a machine preceded by an infinite random access buffer . 61
     9.1.2 Each $m_i$ is a unit capacity machine or buffer .................... 63
  9.2 $N$ Parts and One Resource .............................................. 64

10 Evaluating Gains of Synchronous Matrix Signal Flow Graphs .......... 68
  10.1 Synchronous MSFGs - Preliminary Definitions ...................... 69
  10.2 Regular vs. Synchronous MSFGs ....................................... 71
  10.3 Topological Methods for Evaluating Gains of Synchronous MSFGs .... 72
     10.3.1 Basic Graph Reduction Rules ...................................... 72
     10.3.2 Return Loop Method ............................................... 74
  10.4 Application of Synchronous MSFGs ................................... 77

11 Applications of the Model .................................................. 81
  11.1 Job Shop ................................................................. 81
     11.1.1 The Model .......................................................... 82
     11.1.2 Makespan Computation ............................................. 83
     11.1.3 Scheduling Problem Formulation .................................. 83
     11.1.4 Numerical Example ................................................ 84
  11.2 Cyclic Permutation Flow-Shop ......................................... 86
     11.2.1 The Model .......................................................... 86
     11.2.2 Scheduling Problem Formulation .................................. 89

12 Properties of Inverse Monge Matrices in Max-Plus Algebra ............... 90
  12.1 I/O Graph Representation of Matrices in Max-Plus Algebra ............ 91
  12.2 Max-Plus Algebraic Multiplication of Inverse Monge Matrices .......... 93
  12.3 Computing an Eigenvector of an Inverse Monge Matrix .................. 96
12.3.1 Computing the Metric Matrix ........................................ 97
12.3.2 An Algorithm for Computing an Eigenvector ................. 101
12.4 Monge Property in Modeling of Cyclic Permutation Flow Shops . . . . . . 102

13 Conclusion ................................................................. 107

14 Future Work .............................................................. 109

References ................................................................. 111
List of Figures

3.1 Two disjoint paths \{p_1, p_2, p_3\} and \{q_1, q_2, q_3, q_4\} on the boundary of a convex polygon. The Euclidian distance matrix \([A]_{i,j} = d(p_i, q_j)\) fulfills the Monge property. .......................................................... 24

5.1 Block representation of a manufacturing system: (a) describes block representation of a manufacturing system; and, (b) illustrates flow of parts and resources through the block ...................................................... 33

5.2 Interconnection of manufacturing blocks. An illustrative example. .......... 35

5.3 Interconnection of blocks: (a) part-flow interconnection and (b) machine-flow interconnection. .......................................................... 35

5.4 Composition of blocks .......................................................... 38

5.5 Obtaining \(Q\) and \(R\) from Figure 5.2. .......................................... 42

6.1 Single machine manufacturing single part. ........................................ 44

6.2 Assembly and disassembly machines ............................................. 45

6.3 Buffer models .................................................................. 46

6.4 Machine with an infinite buffer in front of it processing a part ............... 47

7.1 Interconnection diagram for job shop example .................................... 49

7.2 Detailed block diagram of the job shop .......................................... 49

8.1 Part-based approach ................................................................ 54

8.2 Machine-based approach .......................................................... 55

8.3 Block representation of a cyclic system .......................................... 56

9.1 Model of a part processed by a set of resources. .............................. 60

9.2 Model of a part processed by a set of machines: each \(m_i\) is a machine preceded by an infinite buffer. .............................................. 62

9.3 Model of a part processed by a set of machines: each \(m_i\) is a unit capacity machine or buffer. .................................................. 63

9.4 Model of a resource processing a set of parts. ................................... 67

10.1 Synchronous MSFG representation of (10.1) .................................. 70

10.2 Removal of a self-loop. ............................................................ 72

10.3 Synchronous MSFG representation of the basic manufacturing block .... 73

10.4 An illustration of the basic reduction rules for regular and synchronous MSFGs. 74

10.5 Given the synchronous MSFG in Figure 10.1, this figure illustrates the computation of node factors on the path \([x_1, x_2, x_3, x_4, x_5]\) for (a) node \(x_2\), (b) node \(x_3\), and (c) node \(x_4\). .................................................. 77

10.6 Model of a unit capacity machine processing a set of parts. ............... 78

10.7 Synchronous MSFG representation of a permutation flow shop with 3 machines 79
11.1 Model of the cyclic permutation flow-shop ....................... 88

12.1 I/o graphs and their series composition. .......................... 92
12.2 A $\Xi$ i/o graph .................................................... 95

14.1 The concept of entities competing for resources. ................. 110
1 INTRODUCTION

A Discrete Event System (DES) is a system, which is characterized by a set of states and a set of events [25]. Events cause the DES to change its state at discrete time instants. In contrast to continuous time systems, which are time-driven, the evolution of DESs in time depends only on the occurrence of discrete events. Examples of DES can be found in various man-made systems such as communication and transportation networks, computer, software and manufacturing systems [8].

This work focuses on deterministic discrete-event manufacturing systems. A manufacturing system consists of a set of resources performing operations on a set of parts. Typical events occurring in a manufacturing system include arrival of a part, completion of a finished part and removal of a finished part from the system. A deterministic manufacturing system is the one which is conflict- or choice free [51], therefore it is assumed that the routes of all the parts are established, the sequences of parts on the machines are known and the processing times are fixed. Hence, the future states of a deterministic system are uniquely determined by the present state and external inputs.

DES theory is mainly concerned with creating DES models. Without such a model it would be impossible to design and control DESs [25]. The design of a manufacturing system usually includes optimization of various system parameters, such as the throughput rate at which each part is produced or the system’s makespan (defined as the time span between the start and the finish of production of a sequence of parts). It has been shown that this is an NP hard problem. Most designers rely on heuristics, e.g., genetic algorithms, simulated annealing, branch and bound searches, neural networks, etc., to solve these problems. An efficient method (i.e., a modeling framework) to calculate the system characteristics will allow the heuristics to be either applied to larger problems or increase the number of iterations that are explored.
This work develops a new hierarchical modeling framework for deterministic manufacturing systems. A block diagram type of model is proposed. A manufacturing system is represented as a network of subsystems. Each subsystem is modeled as a block with three inputs and three outputs. The blocks in the block diagram are interconnected through a) part-flow interconnections, which specify flow of parts through the diagram, and b) resource-flow interconnections, which specify flow of resources through the diagram. The model is expressed as a system of simultaneous linear matrix equations in max-plus algebra. Alternatively, the model is graphically represented by a block diagram over max-plus algebra, or, equivalently, by a matrix signal flow graph over max-plus algebra. On the one hand, performance of the model can be evaluated by solving the system of simultaneous matrix equations in max-plus algebra. On the other hand, since the model is based on block diagrams, it is possible to use topological methods for performance evaluation. This work presents new topological methods for evaluating gains of matrix signal flow graphs over max-plus algebra.

When modeling a cyclic permutation flow shop, it was discovered that its system matrix fulfills the so-called inverse Monge property. Many problems admit an easy solution if its underlying data fulfills the inverse Monge property [6] and it turned out that cyclic permutation flow shops are no exception. This work presents some new results for inverse Monge matrices in max-plus algebra, such as an efficient algorithm for computing an eigenvector of an inverse Monge matrix.

The dissertation is organized as follows. Chapter 2 provides a review of the existing modeling approaches for discrete manufacturing systems. Chapter 3 discusses limitations of the existing modeling approaches and briefly describes how the proposed model addresses these limitations. In Chapter 4 an overview of the max-plus algebra is provided. The new block-diagram based modeling approach is described in Chapter 5. In Chapter 6 the models of basic manufacturing blocks, like the model of a unit capacity machine
processing a part, are derived. An application of the model is demonstrated by a numerical example in Chapter 7. The model provides a number of analytical tools for performance evaluation, scheduling and design of manufacturing systems as described in Chapter 8. In Chapter 9 it is shown how the model of a part can be obtained by stacking basic blocks horizontally and how the model of a machine can be obtained by stacking basic blocks vertically. The model can be described by synchronous matrix signal flow graphs, hence in Chapter 10 topological methods for evaluating gains of synchronous matrix signal flow graphs are developed. Chapter 11 demonstrates an application of the proposed model to job shops and cyclic permutation flow shops. New results for inverse Monge matrices in max-plus algebra are provided in Chapter 12. In Chapter 13 conclusions are drawn and directions for future work are discussed in Chapter 14.
2 Literature Review

According to [8], there are two different types of models of DES, namely untimed models and timed models. In untimed models, the system’s evolution is merely viewed as a sequence of states. In timed models, a sequence of states is assigned the time instants at which state transitions take place. Timed models can often be obtained from untimed models by introducing time. In this chapter, we concentrate on timed modeling frameworks of DES, which allow for performance evaluation of the system. The modeling formalisms include timed automata, timed Petri nets and its subclass of timed event graphs, queueing networks, discrete event simulation and max-plus algebra. A brief discussion of the approaches follows.

2.1 Automata Modeling Formalism

DES is characterized by a set of states $X$ and a set of events $E$. The set of events cause the DES to change its state at discrete time instants [25]. A language is a tool for describing the behavior of a DES. The set $E$ can be thought of as an alphabet of the system, event sequences can be thought of as strings (or words) and a language can be thought of as a set of strings. The language specifies admissible sequences of events that DES is capable of generating [8].

An automata is a mathematical formalism that provides a way of representing a language according to well-defined rules. It is defined using $X$, $E$, transition function $f : X \times E \rightarrow X$, initial state $x_0$ and marked state set $X_m$. A detailed description of the automata modeling formalism can be found in [8, Ch.2].

The automata can be represented with a state transition diagram. Automata can be composed by product or parallel composition. Composition of a large number of individual automata, however, can lead to state space explosion [8].
A timed automata is an automata enhanced with a clock structure [8, Ch.5]. Timed automata have been applied to scheduling of manufacturing systems, where the scheduling problem translates into finding the shortest path in the timed automation (e.g. [1, 2]).

2.2 Timed Petri Nets and Timed Event Graphs

Petri nets are a graphical and mathematical modeling tool for description and analysis of DESs. The concept of Petri nets was developed by C.A. Petri who introduced it to the public in 1962 in his dissertation work. Since then a large number of analysis techniques has been developed for studying Petri nets [40].

Event graphs (also called marked graphs) are a subclass of Petri nets where each place has exactly one incoming and one outgoing arc. Event graphs are deterministic by nature - they can model activities in concurrent choice-free discrete event systems [51].

Timed event graphs (TEGs) can be used to model timing behavior of dynamical deterministic manufacturing systems (e.g. [3, 10]). A TEG representation for dynamical deterministic manufacturing system can be obtained as follows:

- manufacturing operations are modeled by places in a TEG;
- durations of operations are modeled by holding times assigned to places;
- events are modeled by transitions;
- occurrence of an event is modeled by firing its corresponding transition;
- precedence relations among the events are modeled by directed arcs;
- state of the system is represented by marking of TEG.

A directed graph consists of a set of nodes and a set of weighted arcs connecting the nodes. Directed graphs (also called precedence constraint graphs) can model precedence constraints among events in deterministic manufacturing systems. Nodes represent
system’s events and arcs represent system’s operations. Direction of arcs models order of operations in the system. Arcs’ weights model durations of operations associated with the arcs. Topologically, directed graphs are identical to TEGs that have only one token. However, TEGs with multiple tokens provide more compact representation of the system than directed graphs.

TEGs and directed graphs allow for path based approaches to studying performance characteristics of deterministic DES. For example, Karp’s algorithm [30] can be used to evaluate cycle time of periodic deterministic manufacturing systems modeled with directed graphs or TEGs. Levner [35] was the first who used path-based approach to calculate the makespan of a permutation flow shop system. McCormick et al. [38] used directed graph model of a cyclic permutation flow shop to derive effective algorithm for calculating its steady-state cycle time. Lee and Posner [34] studied performance of cyclic job shops, where the minimum cycle time was identified as a circuit measure in a directed graph.

Other performance evaluation and optimization tools for TEGs and directed graphs include integer/linear programming (e.g. [12, 24, 29, 39, 45]) and max-plus algebra. Max-plus algebra is introduced in Section 2.5 and its theoretical background is provided in Chapter 4.

Some of limitations of TEG-based modeling approaches include:

- A graphical model has to be obtained before the system can be analyzed.

- It is not easy to study what happens when system configuration changes, i.e. the model is not hierarchial. For example, adding an extra machine or buffer may require re-deriving the entire model from scratch. There is no known technique for composing a set of interacting manufacturing systems into one aggregate system.
2.3 Queueing Networks

A manufacturing system can be modeled as a network of queues, where each machine together with a buffer in front of it is represented by a queue. Usually queueing networks are used to evaluate long term (steady-state) performance characteristics of stochastic manufacturing systems such as average number of parts in buffers, average throughput, etc. (e.g. [4, 5, 42]).

However, a class of queueing networks, called max-plus linear queueing networks (i.e. queueing networks, in which timing of events can be described by equations in the max-plus algebra), can be used to model deterministic manufacturing systems. Krivulin [31, 32] and Heidergott [21] studied such queueing networks. Krivulin [31, 32] applied max-plus algebra to model sample queueing networks which have only one class of items. Heidergott [21] studied necessary and sufficient conditions for max-plus linearity of queueing systems. He has shown that for queueing network to be max-plus linear, all the items must follow the same route through the nodes. This means that max-plus linear queueing networks are limited to modeling deterministic manufacturing systems, in which all parts follow the same route through the machines (e.g. flow-shops).

2.4 Discrete Event Simulation

A typical system model consists of mathematical equations describing the behavior of the system. Discrete event simulation, on the other hand, provides numerical evaluation of the system model. Discrete event simulation is used for cases when "real world" systems are too complex to analyze mathematically, i.e. when analytical solution for a DES model is too hard to obtain or when a system does not conform to some assumptions we make in order to simplify the model [8].

Discrete event simulation is often applied to design and performance evaluation of manufacturing systems, see for example [16, 49]. There exist various software for discrete
event simulation, such as Arena and ProModel [48]. Compared to other modeling approaches, discrete event simulation can be computationally more expensive and it does not provide equations needed to analyze and predict system’s behavior.

2.5 Max-Plus Algebra

The max-plus algebra has only two operations, namely maximization denoted by \( \oplus \) and addition denoted by \( \otimes \). Max-plus algebra is an example of a mathematical structure called commutative and idempotent semiring or dioid [23]. This algebra is linear. What makes the max-plus algebra an attractive tool for modeling of manufacturing systems is the fact that event timing dynamics of any deterministic manufacturing system can be expressed by a set of linear equations in the max-plus algebra. Theoretical framework of the max-plus algebra (which is briefly described Chapter 4) provides tools to study performance characteristics and to analyze behavior of manufacturing systems.

As it was mentioned earlier, the equations describing event timing in either (a) timed event graphs or (b) max-plus linear queueing networks can always be written in terms of max-plus algebra. Event timing equations can also be obtained directly from system specifications. Hence, the existing methods for obtaining a max-plus algebraic model of a deterministic manufacturing system can be roughly categorized into three cases

- Max-plus algebraic model is obtained from timed event graph or from directed graph representation of the system. This is the most widely used approach, see for example [3, 10, 33].

- Max-plus algebraic model is obtained from max-plus linear queueing network representation of the system. See, for example [21, 31, 32].

- Max-plus algebraic model is obtained directly from the system’s specifications. This approach was used by Doustmohammadi et. al [13, 14]. They introduced the notion
of interconnection matrices, which describe precedence relation among operations. Their approach was applied only to modeling of generalized dynamical (cyclic) flow-shops.
3 Problem Statement

This chapter summarizes limitations of the existing modeling approaches and then outlines objectives and contributions of this work.

3.1 Limitations of Existing Models

In summary, the main limitations of the existing models of deterministic manufacturing systems are listed below.

- Max-plus linear queueing networks are limited to modeling manufacturing systems where all parts follow the same route through the machines.

- Composition of large number of individual automata can lead to state space explosion.

- TEG and directed graph models are not modular. They do not offer the simplicity of composition of systems with transfer function block diagrams found in classical control system models. In addition, if the goal is to get a max-plus algebraic model of the system from TEG or directed graph model, then there is an extra step of obtaining TEG or directed graph model first. Frequently, the complexity of a real DES leads to a large and complex Petri net. Analyzing such nets is practically very difficult and time consuming. Therefore, availability of a systematic approach to the Petri net design is highly desirable. The substance of such approaches is model reduction and composition techniques.

- Discrete event simulation does not provide equations needed to analyze and predict system’s behavior.
• Direct max-plus algebraic modeling approach proposed by Doustmohammadi does not offer simplicity of block diagrams and it is only applied to generalized cyclic flow-shops.

The dissertation research aims at addressing these limitations as stated in the next section.

3.2 Objectives and Contributions

What is missing in the existing modeling approaches of manufacturing systems is the block diagram type of model. It is well known that block diagrams, such as state space block diagrams and transfer function block diagrams, are widely used in control theory to model the behavior of continuous-time systems. A transfer function block diagram has four basic elements: block, line segment, pick-off point and summing node. A block may model a controller, a sensor, or a whole plant. In a block diagram external input and output variables are connected to blocks by connection lines. An output of a block may also be connected to an input of another block. The interconnection of components (blocks) can be found by following the paths of signal flow along the connecting lines. One of the advantages of the transfer function representation is the simplicity of the algebraic relations between the subsystem or component transfer functions [15].

In classical control theory transfer function block diagrams may be reduced by systematically applying interconnection conventions of variables. For example, by systematic block reduction a complete system can be reduced to a single transfer function block, which provides necessary relations between system’s inputs and outputs. Some immediate variables may disappear in the simplification process. Block diagram reduction procedure may be difficult for complex systems with many loops, so there is also a Mason’s Gain Rule for signal flow graphs (a signal flow graph is essentially a simplified notation for a block diagram, i.e. see [15, p. 18]), which provides the relations between variables without any required reduction.
The contributions of this work are listed below.

3.2.1 Hierarchical Model of Manufacturing Systems

A block diagram type of model for deterministic manufacturing systems is developed. A block can be a machine queue, a part or a factory. Each block has the same input-output structure with three inputs and three outputs. The blocks in the block diagram are interconnected through a) part-flow interconnections, which specify flow of parts through the diagram, and b) resource-flow interconnections, which specify flow of resources through the diagram. The model is hierarchial – it is shown how a network of blocks can be combined into one block that has the same input-output structure. Mathematically, the model is described by a set of simultaneous linear equations in max-plus algebra. To abstract from the number of inputs, outputs and states, the variables are expressed as vectors and a system of max-plus algebraic equations is written in matrix form. Theoretical knowledge in max-plus algebra attained over the past several decades provides basis for analysis, design and control of manufacturing systems. The model is obtained directly from system description, i.e. it does not require first deriving TEGs, directed graph or queueing network model of the system. The model can be used for: (a) performance evaluation, (b) deadlock detection, (c) structural analysis, (d) scheduling, (e) design, and (f) control of manufacturing systems.

3.2.2 Topological Methods for Performance Evaluation

One the one hand, the model is algebraically described by a system of simultaneous linear matrix equations in max-plus algebra. On the other hand, the model is topologically portrayed by a matrix block diagram over max-plus algebra or, equivalently, a matrix signal flow graph (MSFG) over max-plus algebra. MSFG is a simplified notation for matrix block diagram. Gain of a path in a max-plus algebraic MSFG is the solution to the system of simultaneous matrix equations in max-plus algebra. The problem of finding
gain of a path in MSFG is directly related to performance evaluation of the system. Hence, topological methods for performance evaluation of the model may be used instead of purely algebraic methods.

Mason’s gain formula is a topological procedure used to calculate the gain of a scalar signal flow graph in regular algebra. In general, Mason’s gain formula cannot be used to calculate the gain of a MSFG, mainly because matrix multiplication is non-commutative. There exist several topological methods for evaluating gains of MSFGs in ring algebra [46, 47]. However there is no known technique for obtaining gains of MSFGs in max-plus algebra.

In this work, it is shown that the existing topological methods for evaluating gains on MSFGs in regular algebra can be extended to MSFGs in max-plus algebra. The main challenge in applying the existing topological procedures to MSFGs in max-plus algebra is due to the fact that the inverse of a matrix is not defined in max-plus algebra.

### 3.2.3 Properties of Inverse Monge Matrices

An $n$ by $m$ matrix $A$ is called a Monge matrix, if

$$a_{ij} + a_{kl} \leq a_{il} + a_{kj}, \text{ for all } i < k, j < l.$$  

Similarly, $A$ is an inverse Monge matrix, if

$$a_{ij} + a_{kl} \geq a_{il} + a_{kj}, \text{ for all } i < k, j < l.$$  

The above properties can also be written in terms of max-plus algebra, i.e. $A \in \mathbb{R}^{n \times m}_{\min}$ is a Monge matrix if

$$a_{ij} \oplus a_{kl} \geq a_{il} \oplus a_{kj}, \text{ for all } i < k, j < l.$$  

$A \in \mathbb{R}^{n \times m}_{\max}$ is an inverse Monge matrix if

$$a_{ij} \oplus a_{kl} \leq a_{il} \oplus a_{kj}, \text{ for all } i < k, j < l.$$
Monge and inverse Monge property arises in many practical applications, such as transportation, traveling salesman, dynamic programming, economic lot sizing and path problems [7]. Consider the following example. Let \( \{p_1, p_2, \ldots\} \) and \( \{q_1, q_2, \ldots\} \) be two disjoint paths on the boundary of a convex polygon as shown in Figure 3.1, and let \( d(p_i, q_j) \) denote the Euclidian distance between \( p_i \) and \( q_j \), then \( [A]_{i,j} = d(p_i, q_j) \) is a Monge matrix [6].

![Figure 3.1: Two disjoint paths \( \{p_1, p_2, p_3\} \) and \( \{q_1, q_2, q_3, q_4\} \) on the boundary of a convex polygon. The Euclidian distance matrix \( [A]_{i,j} = d(p_i, q_j) \) fulfills the Monge property.](image)

Many discrete optimization problems admit an easy solution if they involve Monge (or inverse Monge) matrices. For example, a class of traveling salesman problems can be solved very efficiently if its cost matrix is Monge [7], or the computation of max-plus algebraic eigenvalue of a matrix becomes trivial if the matrix fulfills the inverse Monge property [18].

In Section 11.2, a cyclic permutation flow shop system is modeled using the approach developed in this work. It is shown that the system’s performance characteristics, namely the steady-state period of the system and the steady-state periodic behavior of the system, can be obtained by computing the eigenvalue and eigenvectors of the system matrix \( \mathbf{D} \), respectively. For general irreducible \( n \times n \) matrices in max-plus
algebra, no better than $O(n^3)$ algorithms are known for computing its max-plus algebraic eigenvalue and eigenvectors [18]. In this work properties of inverse Monge matrices are studied in detail. The study is motivated by our discovery that $\mathbf{D}$ is an inverse Monge matrix [26]. This means that performance characteristics of cyclic permutation flow-shops can be computed very efficiently. For example, Gavalec and Plavka [18] have shown that the eigenvalue of an inverse Monge matrix can be computed in just $O(n)$ time.

This work presents the following new contributions to the theory and application of inverse Monge matrices:

- it is proven that the system matrix for permutation flow shops fulfills the inverse Monge property;

- it is proven that the class of inverse Monge matrices is closed under max-plus algebraic multiplication; and

- an efficient $O(n^2)$ algorithm for computing an eigenvector of an inverse Monge matrix is presented.
4 Max-Plus Algebra

This section provides background information on max-plus algebra. A comprehensive review of the max-plus algebra can be found in [23], from which most of the notation introduced in this chapter is adopted.

4.1 Basics

Define $\varepsilon = -\infty$ and $\mathbb{R}_{\text{max}} = \{\mathbb{R} \cup \varepsilon\}$, where $\mathbb{R}$ is the set of real numbers. The two max-plus algebraic operations, $\oplus$ and $\otimes$, are defined as follows:

$$a \oplus b = \max(a, b) \quad a \otimes b = a + b,$$

for elements $a, b \in \mathbb{R}_{\text{max}}$.

Max-plus algebra is an example of algebraic structure called dioid. Operation $\oplus$ has null element, $\varepsilon$, since $a \oplus \varepsilon = a$. Similarly operation $\otimes$ has unit element, $e = 0$, as $a \otimes e = a$.

Max plus algebra is extended to matrices in the same way as conventional algebra but with $+$ replaced by $\oplus$ and $\times$ replaced by $\otimes$. Let $\mathbb{R}_{\text{max}}^{m \times n}$ denote the set of all $n \times m$ matrices with elements in $\mathbb{R}_{\text{max}}$. For matrices $A, B \in \mathbb{R}_{\text{max}}^{m \times n}$, the sum of matrices $A \oplus B$ is defined as

$$[A \oplus B]_{i,j} = a_{i,j} \oplus b_{i,j},$$

for $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, n\}$. The matrix product $A \otimes B$ is defined as

$$[A \otimes B]_{i,j} = \bigoplus_{k=1}^{n} a_{i,k} \otimes b_{k,j} = \max_{k \in \{1, 2, \ldots, n\}} (a_{i,k} + b_{k,j}),$$

for $A \in \mathbb{R}_{\text{max}}^{m \times n}$, $B \in \mathbb{R}_{\text{max}}^{n \times l}$, $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, l\}$.

We say that an $n \times m$ matrix $A$ exists if and only if $A \in \mathbb{R}_{\text{max}}^{n \times m}$. Analogous to conventional algebra $\otimes$ is assumed precedence over $\oplus$ and if it is clear that the $\otimes$ symbol is used it is sometimes omitted, i.e. $A \otimes BC$ should be understood as $A \oplus (B \otimes C)$.

Eigenvalues and corresponding eigenvectors for square matrices in max-plus algebra are defined in just the same way as in conventional algebra as described in Section 4.4. In
max-plus algebra there is a close relation between matrices and directed graphs. This relation is very useful in understanding spectral theory of matrices in max-plus algebra, i.e. eigenvalue and eigenvector problem and its extensions. The following introduces the concept of communication graph associated with a square matrix in $\mathbb{R}^{n \times n}_{\text{max}}$.

### 4.2 Communication Graph

A directed graph $G$ is an ordered pair $(N, E)$ where $N$ is a finite set of nodes and $E$ is a set of ordered pairs of nodes called arcs. Thus a pair $(i, j)$ denotes an arc directed from node $i$ to node $j$. In a weighted directed graph each arc is assigned a weight in $\mathbb{R}$. Hereafter a weighted directed graph will simply be referred to as graph.

**Definition 4.2.1.** To any square matrix $A \in \mathbb{R}^{n \times n}_{\text{max}}$, we can associate a graph $(N, E)$, which is denoted by $H(A)$ and is called the communication graph of $A$. $H(A)$ has $n$ nodes and its set of nodes is given by $N(A) = \{1, 2, \ldots, n\}$. For any two nodes $i, j \in N(A)$, there is an arc with weight $a_{ji}$ if and only if $a_{ji} \neq \epsilon$.

A path $p$ in $H(A)$ can be denoted as a sequence of its arcs

$$p = (i_1, i_2), (i_2, i_3), \ldots, (i_{r-1}, i_r),$$

as a sequence of its nodes

$$p = (i_1, i_2, \ldots, i_r),$$

and as a mixed sequence of its nodes, arcs and sub-paths

$$p = (i_1, i_2) \xrightarrow{p_1} (i_k, i_{k+1}) \xrightarrow{p_2} i_l \xrightarrow{p_3} i_r,$$

where $\xrightarrow{p_1}$ denotes a sub-path of $p$ from $i_2$ to $i_k$, $\xrightarrow{p_2}$ denotes a sub-path of $p$ from $i_{k+1}$ to $i_l$ and $\xrightarrow{p_3}$ denotes a sub-path of $p$ from $i_l$ to $i_r$.

The length of a path is the number of arcs in a path and it is denoted by $|p|$. The weight of the path denoted by $|p|_w$ is defined as a sum of weights of all the arcs
constituting the path. The average weight of a path $p$ is given by $|p|_w/|p|$. A path is called elementary if, restricted to the path, nodes $i_k$ and $i_l$ are different for $k \neq l$. A circuit $\gamma = (i_1, i_2, \ldots, i_r)$ is a path in which $i_1 = i_r$. A circuit is called elementary if, restricted to the circuit, nodes $i_k$ and $i_l$ are different for $k \neq l$ (except for $i_1 = i_r$). For circuits, the notions of length, weight and average weight are defined similarly as for paths. A circuit consisting of a single arc, from a node to itself, is called a self-loop.

A graph $(N, E)$ is called strongly connected if for any two nodes $i, j \in N$ there exists a path from $i$ to $j$. A matrix over $\mathbb{F}_{\text{max}}^{n \times n}$ is called irreducible if its communication graph is strongly connected.

4.3 Definitions and interpretations of $A^{\otimes k}$, $A^+$ and $A^*$

Let $A \in \mathbb{F}_{\text{max}}^{n \times n}$ be a square matrix in max-plus algebra. $A$ in $n^{th}$ power is defined by

$$A^{\otimes n} = A \otimes A \otimes \ldots \otimes A.$$ 

Let $P(i, j; k)$ denote the set of all paths in $H(A)$ from $i$ to $j$ of length $k \geq 1$. Then for all $k \geq 1$ we have

$$[A^{\otimes k}]_{ij} = \max \{|p|_w : p \in P(i, j; k)\}$$

Define

$$A^+ = A \oplus \ldots \oplus A^{\otimes k} \oplus A^{\otimes (k+1)} \oplus \ldots = \bigoplus_{k=1}^{k=\infty} A^{\otimes k}.$$ 

The element of $[A^+]_{ij}$ gives the maximal weight of any path in $H(A)$ from $j$ to $i$ [23, p.31].

Define $A_\lambda = -\lambda(A) \otimes A$, i.e. $[A_\lambda]_{ij} = a_{ij} - \lambda(A)$. Then $A_\lambda^+$ is well defined [23], i.e.

$$A_\lambda^+ = \bigoplus_{k=1}^{k=n} A^{\otimes k}.$$ 

Define Kleene star operator on $A \in \mathbb{F}_{\text{max}}^{n \times n}$ denoted by $A^*$ as

$$A^* = A^{\otimes 0} \oplus A^{\otimes 1} \oplus \ldots \oplus A^{\otimes \infty} = \bigoplus_{k=0}^{k=\infty} A^{\otimes k}.$$
where $A^{00} = E$ and $E \in \mathbb{R}^{n \times n}_{\text{max}}$ refers to identity matrix which has $e$’s on the main diagonal and $\varepsilon$’s elsewhere. Note that

$$A^* = E \oplus A^+.$$ 

According to [20] $A^*$ can be computed in at most $O(n^3)$ time using the Floyd-Warshall algorithm. $A^*$ can be used in solving linear equations as shown in the next theorem.

**Theorem 4.3.1.** [23, Theorem 2.10] $x = A^* \otimes b$ solves the equation $x = A \otimes x \oplus b$, provided that $A^*$ exists.

### 4.4 Eigenvectors and Eigenvalues

**Definition 4.4.1.** The problem of finding a value $\lambda(A) \in \mathbb{R}_{\text{max}}$ and a vector $v \in \mathbb{R}^n_{\text{max}}$ that contains at least one finite element such that

$$A \otimes v = \lambda(A) \otimes v$$

is called an eigenproblem of $A \in \mathbb{R}^{n \times n}_{\text{max}}$. We call $\lambda(A)$ an eigenvalue and $v$ an associated eigenvector.

The problem of finding eigenvalues and eigenvectors of $A$ in max-plus algebra has been studied in [3, 9, 17, 20, 30, 36, 50], to name a few. For computing eigenvalues of $A$, Karp’s algorithm [30] has the best worst-case performance $O(n^3)$. As far as computing eigenvectors, no better algorithm than $O(n^3)$ is known for matrices of general type [19]. The following describes some graph-theoretical results related to eigenvalues and eigenvectors, which will be used in Chapter 12.

#### 4.4.1 Eigenvalue

Graphical interpretation of eigenvalue of irreducible matrix follows. An elementary circuit $y$ in $H(A)$ is called critical if its average weight is maximal over all circuits in
\( H(\mathbf{A}) \). By Baccelli [3], \( \lambda(\mathbf{A}) \) equals the average weight of a critical circuit in \( H(\mathbf{A}) \), i.e.

\[
\lambda(\mathbf{A}) = \max_{\gamma \in C(\mathbf{A})} \frac{|\gamma|w}{|\gamma|},
\]

(4.1)

where \( C(\mathbf{A}) \) is the set of all elementary circuits in \( H(\mathbf{A}) \).

### 4.4.2 Eigenvectors

Let \([\mathbf{A}]_{ij} = a_{ij} - \lambda(\mathbf{A})\), where \( \lambda(\mathbf{A}) \) is the eigenvalue of an irreducible square matrix \( \mathbf{A} \). The following result was first shown in [20].

**Theorem 4.4.2 ([20]).** Let \( K \) be a set of integers such that \([\mathbf{A}]^+_i k = e\), for any \( k \in K \).

Every column \( k \in K \) in \( \mathbf{A}^+_i \) is a (fundamental) eigenvector of \( \mathbf{A} \). Moreover, every eigenvector of \( \mathbf{A} \) can be expressed as a linear combination of fundamental eigenvectors.

Let \([\mathbf{A}]_{i j}^+\) denote the \( n \)-th column of a matrix \( \mathbf{A} \). Let \( k \in K \), then an eigenvector of \( \mathbf{A} \), \( \mathbf{v} \), is given by the following expression

\[
\mathbf{v} = [\mathbf{A}^+_i]_{e k}.
\]

(4.2)

Let \( P(i, k) \) denote the set of paths in \( H(\mathbf{A}, i) \) from node \( i \) to node \( k \in K \), then

\[
[\mathbf{v}]_i = [\mathbf{A}^+_i]_{i k} = \max \{|p|_w : p \in P(k, i)\},
\]

(4.3)

In other words, \([\mathbf{v}]_i\) yields the maximum weight of a path from \( k \) to \( i \) in \( H(\mathbf{A}, i) \).

Matrix \( \mathbf{A}^+_i \) can be found in \( O(n^3) \) time using the Floyd-Warshall algorithm [11], therefore a complete set of fundamental eigenvectors can be found in at most \( O(n^3) \) operations.
5 The Model

This chapter is organized as follows. First, a general block representation of a deterministic manufacturing system is introduced. Then, an interconnection mechanism for combining several manufacturing subsystems (i.e. blocks) into one large manufacturing system (i.e. block diagram) is described. Finally, a composition formula for reducing a manufacturing block diagram into one block is derived. Some of the work presented in this chapter and the following chapter was previously published in the conference proceedings [27, 28].

The following standard notation is used in this chapter and thereafter. For a positive integer $K$, define $K = \{1, 2, \ldots, K\}$. Let $s$ be an ordered set or a vector. Then $|s|$ gives the number of elements in $s$. The $i$-th element of $s$ is denoted by $[s]_i$, for any $i \in |s|$.

5.1 General Block

Consider a manufacturing system. In order to operate, the system requires a set of parts and a set of resources. After the system is done with the parts and the resources, they are released by the system. Let $m$ denote an ordered set of system’s resources, such as machines, buffers, etc. Let $n^{in}$ be the ordered set of parts that enter the system and let $n^{out}$ be the ordered set of parts that leave the system. The order of elements in either $m$, $n^{in}$ or $n^{out}$ can be chosen arbitrary. For $k \in \{1, 2, \ldots, |m|\}$, let $[m]_k$ denote the $k$-th resource in the set $m$. Similarly, for $i \in \{1, 2, \ldots, |n^{in}|\}$ and $j \in \{1, 2, \ldots, |n^{out}|\}$, let $[n^{in}]_i$ and $[n^{out}]_j$ denote the $i$-th part in $n^{in}$ and $j$-th part in $n^{out}$, respectively. If the manufacturing process involves part assembly or disassembly then $n^{in} \neq n^{out}$ because during assembly several parts are needed to create a new part and during disassembly a single part is disassembled into several new parts. If there are no assembly and disassembly machines in the system then we can set $n^{in} = n^{out} = n$. 
The system can be modeled by a block with three inputs and three outputs. The inputs, \( u, v \) and \( w \) are defined as

- \( [u]_i \) is the time when \( [n^\text{in}]_i \) becomes available for the system;
- \( [v]_j \) is the time when \( [n^\text{out}]_j \) is removed from the system;
- \( [w]_k \) is the time when \( [m]_k \) becomes available for the system.

The outputs, \( x, y \) and \( z \) are defined as:

- \( [x]_j \) is the time when \( [n^\text{out}]_j \) is ready to leave the system;
- \( [y]_i \) is the time when \( [n^\text{in}]_i \) actually enters the system;
- \( [z]_k \) is the time when \( [m]_k \) is “set free” by the system.

It can be seen that input and output variables are defined with respect to \( m, n^\text{in} \) and \( n^\text{out} \). In particular, \( m \) is associated with \( w, z \); \( n^\text{in} \) is associated with \( u, y \); and \( n^\text{out} \) is associated with \( x \) and \( v \).

It is assumed that the system is deterministic, i.e. the routing of parts through the resources, the processing order of parts on the resources and the processing times of parts on the resources are known and fixed. Then its output can be described in terms of its input by the following equation in the max-plus algebra

\[
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} =
\begin{bmatrix}
  F_{uu} & F_{uv} & F_{uw} \\
  F_{vy} & F_{vy} & F_{yw} \\
  F_{zu} & F_{zv} & F_{zw}
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix}
= F
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix},
\]

where \( F \) is a matrix that describes input-output relation - it is called the system matrix.

The model provides an abstraction of any deterministic manufacturing system by means of block diagram having three inputs and three outputs and system matrix \( F \) as shown in Figure 5.1(a).
The matrix $F$ and $m, n^{in}, n^{out}$ completely describe the model, since variables $u, v, w, x, y, z$ are defined with respect to $m, n^{in}$ and $n^{out}$. Hence, the model $S$ is denoted by a 4-tuple

$$S = (F, m, n^{in}, n^{out}).$$

A block shown in Figure 5.1(b) illustrates flow of parts and resources through the model. The block has two inputs and two outputs. It does not contain information about timing behavior of the system. This block can be used instead of the block shown in Figure 5.1(a) in the case when we are only interested in flow of parts and resources through a network of manufacturing blocks.

![Figure 5.1: Block representation of a manufacturing system: (a) describes block representation of a manufacturing system; and, (b) illustrates flow of parts and resources through the block](image)

5.2 Composition of Blocks

Let $S_c$ be a system composed from a set of $M$ manufacturing subsystems $\{S_1, S_2, \ldots, S_M\}$. Let $m_c, n^{in}_c, n^{out}_c$ be ordered sets of resources and parts associated with $S_c$. Let the inputs and the outputs of $S_c$, namely $u_c, v_c, w_c$ and $x_c, y_c, z_c$, be defined with respect to $m_c, n^{in}_c, n^{out}_c$. 

Each subsystem $S_{i \in M}$ is represented by an equation of the form (5.1) or, specifically,

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} F_{iu,i} & F_{ix,i} & F_{ixw,i} \\ F_{ju,i} & F_{jy,i} & F_{jyw,i} \\ F_{zu,i} & F_{zz,i} & F_{zww,i} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} F_i \\ v_i \\ w_i \end{bmatrix}, \quad (5.2)$$

for $i \in 1, 2, \ldots, M$.

The subsystems $S_1, S_2, \ldots, S_M$ all share the system's parts and resources. It is assumed that there are no delays associated with transportation of parts or resources from $S_j$ to $S_i$ – rather these delays can always be modeled by an appropriate manufacturing block or as part of $S_i$ or $S_j$.

An example illustrating routing of parts and resources through subsystems is given in Figure 5.2. The blocks in the diagram have the form shown in Figure 5.1(b). There are 4 parts $\{n_1, n_2, n_3, n_4\}$ and 3 resources $\{m_1, m_2, m_3\}$. The labeled arrows that interconnect the blocks in the diagram indicate flow of parts and resources through the subsystems. Note that $m = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}^T$, $n_{in} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}^T$ and $n_{out} = \begin{bmatrix} n_1 \\ n_4 \end{bmatrix}^T$. It can be seen that $S_3$ is an assembly operation, in which parts $n_2$ and $n_3$ are assembled by $m_3$ to create a new part $n_4$. Since there is an assembly operation in the system, we have $n_{in} \neq n_{out}$.

Consider part $n$ which enters system $S_i$ from an upstream system $S_j$. Suppose that $n = [n_i^{\text{out}}]_l = [n_i^{\text{in}}]_k$, where indexes $l$ and $k$ point to the location of $n$ in $n_i^{\text{out}}$ and $n_i^{\text{in}}$, respectively. It is said that $[n_i^{\text{out}}]_l$ is routed to $[n_i^{\text{in}}]_k$, which is denoted by $[n_i^{\text{out}}]_l \rightarrow [n_i^{\text{in}}]_k$. Since $n$ becomes available to $S_i$ at the time instance when it is ready to leave $S_j$, we have $[u_j]_k = [x_j]_l$. In addition, the part $n$ is removed from $S_j$ when $n$ enters $S_i$, therefore $[v_j]_l = [y_j]_k$, as shown in Figure 5.3(a).

Likewise, consider a resource $m$, which is first used by $S_j$ and then it is used by $S_i$. Suppose that $m = [m_j]_l = [m_i]_k$, where $l$ and $k$ point to the location of $m$ in $m_j$ and $m_i$, respectively. It is said that $[m_j]_l$ is routed to $[m_i]_k$, which is denoted by $[m_j]_l \rightarrow [m_i]_k$. 


Resource $m$ is available to $S_i$ after $S_j$ is done using $m$, therefore $[w_j]_k = [z_j]_l$, as shown in Figure 5.3(b).

Figure 5.2: Interconnection of manufacturing blocks. An illustrative example.

Figure 5.3: Interconnection of blocks: (a) part-flow interconnection and (b) machine-flow interconnection.
It follows that flow of parts through manufacturing sub-systems is represented by horizontal interconnections (e.g., Figure 5.3(a)). We will refer to this type of interconnections as part-flow interconnections. Likewise, flow of resources through manufacturing sub-systems is represented by vertical interconnections (e.g., Figure 5.3(b)). We will refer to this type of interconnections as resource-flow interconnections.

Routing of parts and resources through the diagram is mathematically represented by means of part-flow and resource-flow interconnection matrices. Define

\[
\mathbf{n}_{\text{in}} = \begin{bmatrix}
\mathbf{n}_{1}^{\text{in}} \\
\mathbf{n}_{2}^{\text{in}} \\
\vdots \\
\mathbf{n}_{M}^{\text{in}}
\end{bmatrix}, \quad \mathbf{n}_{\text{out}} = \begin{bmatrix}
\mathbf{n}_{1}^{\text{out}} \\
\mathbf{n}_{2}^{\text{out}} \\
\vdots \\
\mathbf{n}_{M}^{\text{out}}
\end{bmatrix}, \quad \mathbf{m} = \begin{bmatrix}
\mathbf{m}_{1} \\
\mathbf{m}_{2} \\
\vdots \\
\mathbf{m}_{M}
\end{bmatrix}.
\]

Resource-flow interconnection matrices are defined as:

\[
\left[ Q_{\text{in}} \right]_{i \in \mathbf{m}, j \in \mathbf{m}} = \begin{cases}
\epsilon & \text{if } \left[ \mathbf{m}_{c} \right]_{j} \rightarrow \left[ \mathbf{m} \right]_{i}, \\
\epsilon & \text{otherwise};
\end{cases}
\]

\[
\left[ Q \right]_{i \in \mathbf{m}, j \in \mathbf{m}} = \begin{cases}
\epsilon & \text{if } \left[ \mathbf{m} \right]_{j} \rightarrow \left[ \mathbf{m} \right]_{i}, \\
\epsilon & \text{otherwise};
\end{cases}
\]

\[
\left[ Q_{\text{out}} \right]_{i \in \mathbf{m}, j \in \mathbf{m}} = \begin{cases}
\epsilon & \text{if } \left[ \mathbf{m} \right]_{j} \rightarrow \left[ \mathbf{m}_{c} \right]_{i}, \\
\epsilon & \text{otherwise}.
\end{cases}
\]

Part-flow interconnection matrices are defined as:

\[
\left[ R_{\text{in}} \right]_{i \in \mathbf{n}^{\text{in}}, j \in \mathbf{n}^{\text{in}}} = \begin{cases}
\epsilon & \text{if } \left[ \mathbf{n}^{\text{in}} \right]_{j} \rightarrow \left[ \mathbf{n}^{\text{in}} \right]_{i}, \\
\epsilon & \text{otherwise};
\end{cases}
\]

\[
\left[ R \right]_{i \in \mathbf{n}^{\text{in}}, j \in \mathbf{n}^{\text{in}}} = \begin{cases}
\epsilon & \text{if } \left[ \mathbf{n}^{\text{out}} \right]_{j} \rightarrow \left[ \mathbf{n}^{\text{in}} \right]_{i}, \\
\epsilon & \text{otherwise}.
\end{cases}
\]
\[
[R_{\text{out}}]_{i \in \tilde{n}_{\text{out}}, j \in n_{\text{out}}} = \begin{cases} 
 e & \text{if } [\tilde{n}_{\text{out}}]_j \rightarrow [n_{\text{out}}]_i, \\
 \varepsilon & \text{otherwise}.
\end{cases}
\]

Define
\[
\tilde{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{bmatrix}, \quad \tilde{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{bmatrix}, \quad \tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}.
\]

Similarly define \( \tilde{x}, \tilde{y} \) and \( \tilde{z} \). Define
\[
\tilde{F}_{\text{xu}} = \begin{bmatrix} F_{\text{xu},1} & \varepsilon & \varepsilon \\ \varepsilon & F_{\text{xu},2} & \varepsilon \\ \varepsilon & \varepsilon & \ddots \end{bmatrix}.
\]

Similarly define \( \tilde{F}_{\text{xy}}, \tilde{F}_{\text{xw}}, \tilde{F}_{\text{yu}}, \) etc. Then
\[
\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \tilde{F}_{\text{xu}} & \tilde{F}_{\text{xy}} & \tilde{F}_{\text{xw}} \\ \tilde{F}_{\text{yu}} & \tilde{F}_{\text{yw}} & \tilde{F}_{\text{yw}} \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \tilde{F} \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}.
\] (5.3)

From the definition of the interconnection matrices it follows that
\[
\tilde{u} = R \tilde{x} \oplus R_{\text{in}} u_c, \\
\tilde{v} = R^T \tilde{y} \oplus R_{\text{out}}^T v_c, \quad \text{(5.4)}
\]
\[
\tilde{w} = Q \tilde{z} \oplus Q_{\text{out}} w_c.
\]

Outputs of \( S_c \) can be expressed as
\[
\begin{aligned}
x_c &= R_{\text{out}} \tilde{x}, \\
y_c &= R_{\text{in}}^T \tilde{y}, \\
z_c &= Q_{\text{out}} \tilde{z}.
\end{aligned} \quad \text{(5.5)}
\]
Equations (5.3), (5.4) and (5.5) describe block diagram shown in Figure 5.4 and they can be used to find $F_c$.

Equations (5.4) can be written as

$$
\begin{bmatrix}
\tilde{u} \\
\tilde{v} \\
\tilde{w}
\end{bmatrix}
= \begin{bmatrix}
R & \varepsilon & \varepsilon \\
\varepsilon & R^T & \varepsilon \\
\varepsilon & \varepsilon & Q
\end{bmatrix}
\begin{bmatrix}
\tilde{x} \\
\tilde{y} \\
\tilde{z}
\end{bmatrix}
= \begin{bmatrix}
R_{in} & \varepsilon & \varepsilon \\
\varepsilon & R_{out}^T & \varepsilon \\
\varepsilon & \varepsilon & Q_{in}
\end{bmatrix}
\begin{bmatrix}
u_c \\
v_c \\
w_c
\end{bmatrix}.
$$

(5.6)

Equations (5.5) can be written as

$$
\begin{bmatrix}
x_c \\
y_c \\
z_c
\end{bmatrix}
= \begin{bmatrix}
R_{out} & \varepsilon & \varepsilon \\
\varepsilon & R_{in}^T & \varepsilon \\
\varepsilon & \varepsilon & Q_{out}
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}.
$$

(5.7)
Substituting (5.6) into (5.3) we obtain
\[
\begin{align*}
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z}
\end{bmatrix} &= \begin{bmatrix} F & \varepsilon & \varepsilon \\ \varepsilon & R^T & \varepsilon \\ \varepsilon & \varepsilon & Q \end{bmatrix} \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} + \begin{bmatrix} R_{\text{in}} & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} u_c \\
v_c \\
w_c \end{bmatrix}, \quad (5.8)
\end{align*}
\]

From Theorem 4.3.1 it follows that
\[
\begin{align*}
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z}
\end{bmatrix} &= \begin{bmatrix} \tilde{F} & \varepsilon & \varepsilon \\ \varepsilon & \tilde{R}^T & \varepsilon \\ \varepsilon & \varepsilon & \tilde{Q} \end{bmatrix} \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} + \begin{bmatrix} \tilde{R}_{\text{in}} & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} u_c \\
v_c \\
w_c \end{bmatrix}, \quad (5.9)
\end{align*}
\]

Substituting (5.9) into (5.7) we get
\[
\begin{align*}
\begin{bmatrix}
x_c \\
y_c \\
z_c
\end{bmatrix} &= \begin{bmatrix} R_{\text{out}} & \varepsilon & \varepsilon \\ \varepsilon & R_{\text{in}}^T & \varepsilon \\ \varepsilon & \varepsilon & Q_{\text{out}} \end{bmatrix} \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} + \begin{bmatrix} \tilde{R}_{\text{in}} & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} \tilde{u} \\
\tilde{v} \\
\tilde{w}
\end{bmatrix} \begin{bmatrix} u_c \\
v_c \\
w_c \end{bmatrix}, \quad (5.10)
\end{align*}
\]

Equation (5.10) gives general expression for the system matrix of $S_c$. This proves that any composition of systems represented by (5.1) results in a system that is also represented by (5.1).

Sometimes instead of explicitly specifying $v_c$ it is assumed that jobs are removed from the system as soon as they are ready to leave the system. In other words machines are never blocked from outside of the system. Then $v_c = x_c = R_{\text{out}} \tilde{x}$ and we have
\[
\begin{align*}
\dot{u} &= R \tilde{x} \oplus R_{\text{in}} u_c, \\
\dot{v} &= R^T \tilde{y} \oplus R_{\text{out}}^T v_c = R^T \tilde{y} \oplus R_{\text{out}}^T R_{\text{out}} \tilde{x}, \quad (5.11) \\
\dot{w} &= Q \tilde{z} \oplus Q_{\text{in}} w_c,
\end{align*}
\]
which can be written as

\[
\begin{bmatrix}
\bar{u} \\
\bar{v} \\
\bar{w}
\end{bmatrix} = 
\begin{bmatrix}
R & \epsilon & \epsilon \\
\epsilon & R^T & \epsilon \\
\epsilon & \epsilon & Q
\end{bmatrix}
\begin{bmatrix}
\bar{x} \\
\bar{y} \\
\bar{z}
\end{bmatrix} + 
\begin{bmatrix}
\epsilon & \epsilon & \epsilon \\
\epsilon & Q & \epsilon \\
\epsilon & \epsilon & Q_{in}
\end{bmatrix}
\begin{bmatrix}
u_{c} \\
w_{c}
\end{bmatrix}.
\]

Substituting (5.12) into (5.3) and (5.7) and after some algebraic manipulation it follows that

\[
\begin{bmatrix}
x_{c} \\
y_{c} \\
z_{c}
\end{bmatrix} = 
\begin{bmatrix}
R_{out} & \epsilon & \epsilon \\
\epsilon & R_{in}^T & \epsilon \\
\epsilon & \epsilon & Q_{out}
\end{bmatrix}
\begin{bmatrix}
\tilde{F} \\
\tilde{F} R_{out}^T R_{out} R_{in}^T & \epsilon & \epsilon \\
\epsilon & Q_{in}
\end{bmatrix}
\begin{bmatrix}
u_{c} \\
w_{c}
\end{bmatrix}.
\]

**Example 5.2.1.** Consider a network of three manufacturing blocks, namely

\[
S_{1} = (F_{1}, m_{1}, n_{1}^{in}, n_{1}^{out}) = \left( F_{1}, \begin{bmatrix} m_{1} \\ m_{2} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{2} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{2} \end{bmatrix} \right),
\]

\[
S_{2} = (F_{2}, m_{2}, n_{2}^{in}, n_{2}^{out}) = \left( F_{2}, \begin{bmatrix} m_{2} \\ m_{3} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{3} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{3} \end{bmatrix} \right),
\]

\[
S_{3} = (F_{3}, m_{3}, n_{3}^{in}, n_{3}^{out}) = \left( F_{3}, \begin{bmatrix} m_{3} \\ m_{4} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{3} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{3} \end{bmatrix} \right).
\]

The network of blocks comprise a composite manufacturing block

\[
S_{c} = (F_{c}, m_{c}, n_{c}^{in}, n_{c}^{out}) = \left( F_{c}, \begin{bmatrix} m_{1} \\ m_{2} \\ m_{3} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{2} \\ n_{3} \end{bmatrix}, \begin{bmatrix} n_{1} \\ n_{2} \\ n_{3} \end{bmatrix} \right).
\]

Flow of parts and resources through the network is given by interconnection diagram in Figure 5.2. The goal is to find part-flow and resource-flow interconnection matrices.
We have

\[
\tilde{m} = \begin{bmatrix} m_1 \\ m_2 \\ m_2 \\ m_3 \end{bmatrix}, \quad \tilde{n}^{\text{in}} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_3 \end{bmatrix}, \quad \tilde{n}^{\text{out}} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix},
\]

\[
m_c = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \quad n_c^{\text{in}} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}, \quad n_c^{\text{out}} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}.
\]

From the interconnection diagram in Figure 5.2 it follows that

\[
R^{\text{in}} = \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \\ e & e & e \end{bmatrix}, \quad R = \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \\ e & e & e \end{bmatrix}, \quad R^{\text{out}} = \begin{bmatrix} e & e & e \\ e & e & e \end{bmatrix},
\]

\[
Q^{\text{in}} = \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \\ e & e & e \end{bmatrix}, \quad Q = \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \\ e & e & e \end{bmatrix}, \quad Q^{\text{out}} = \begin{bmatrix} e & e & e \\ e & e & e \end{bmatrix}.
\]

Figure 5.5 illustrates how \(Q\) and \(R\) are obtained.
Figure 5.5: Obtaining Q and R from Figure 5.2.
6 Basic Manufacturing Blocks

In this chapter timing models of basic manufacturing blocks are presented, namely the models of:

1. single resource manufacturing a part;

2. assembly block;

3. disassembly block;

4. unit capacity buffer storing a part;

5. random access buffer with unlimited capacity storing a part.

All models share the generic structure described by (5.1) with three inputs and three outputs.

6.1 Single machine processing single part

Consider machine \( m \) processing part \( n \). Let \( t \) be processing time of \( n \) on \( m \). Suppose that the system is modeled by using equation of the form (5.1) having inputs \( u, v, w \) and outputs \( x, y, z \), which are all scalars because there is only one resource and one part.

The part \( n \) enters the system as soon as both \( m \) and \( n \) are available, therefore

\[
y = u \oplus w.
\]

The part is ready to leave the system as soon as its processing is done on the machine, therefore

\[
x = t(u \oplus w).
\]

The machine is “set free” by the system as soon as \( n \) is removed from the system, therefore

\[
z = v.
\]
Thus, we have

\[
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} =
\begin{bmatrix}
  t & e & t \\
  e & e & e \\
  e & e & e
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix},
\]

(6.1)

Block diagram model of the system is provided in Figure 6.1.

Figure 6.1: Single machine manufacturing single part.

The following develops timing models for assembly and disassembly machines. It will be shown that these models differ slightly from the model of a single machine processing a single part.

6.2 Assembly machine

An assembly machine takes several input parts \( n_{in} \) and produces a single new part \( n_{out} = n \) in its output. Suppose that the system is modeled by using equation of the form (5.1) having inputs \( u, v, w \) and outputs \( x, y, z \). Let \( t \) be assembly time. The block diagram of the assembly machine is provided in Figure 6.2(a). Since the assembly machine needs to wait for all the required parts before it can start processing parts, we have

\[
u = [u]_1 \oplus [u]_2 \oplus \ldots \oplus [u]_N = Yu,\
y = Y^T y,
\]

where \( Y = \begin{bmatrix} e & e & \ldots & e \end{bmatrix} \).
Then the model is described by the following equation

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
Yt & \varepsilon & Yt \\
Y^T & \varepsilon & Y^T \\
\varepsilon & \varepsilon & e
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}.
\]  

(6.2)

6.3 Disassembly machine

Disassembly machine takes one part \( \text{n}_{in} = n \) and outputs several parts \( \text{n}_{out} \). Suppose that the system is modeled by using equation of the form (5.1) having inputs \( u, v, w \) and outputs \( x, y, z \). Let \( t \) be disassembly time. The model of disassembly machine is shown in Figure 6.2(b). After the disassembly process all the output parts are ready to leave the system at the same time, therefore

\[
x = Y^Tx
\]

\[
v = Yv.
\]

The model is described by

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
\varepsilon & Y^T & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}.
\]  

(6.3)

(a) Assembly machine  
(b) Disassembly machine

Figure 6.2: Assembly and disassembly machines
6.4 Unit capacity buffer

McCormick et al. [38] show that a buffer of unit capacity can be represented by a resource having zero processing time for jobs that enter the buffer. Therefore for buffer of unit capacity (6.1) becomes

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
e & e & e \\
e & e & e \\
e & e & e
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix},
\]

(6.4)

because \( t = e \). Block diagram representation of (6.5) is provided in Figure 6.3(a).

![Block diagram of unit capacity buffer](a)

(a) Unit capacity buffer

![Block diagram of unlimited capacity buffer](b)

(b) Unlimited capacity buffer with random access

Figure 6.3: Buffer models

6.5 Random access buffer with infinite capacity

Consider random access buffer with unlimited capacity for storing parts. The buffer is always available to accept parts because of its unlimited capacity, therefore \( w = \varepsilon \) and \( z = \varepsilon \). The part enters the buffer as soon as it becomes available to the buffer, therefore \( y = u \). Also, the part is ready to leave the buffer as soon as it entered the buffer, therefore \( x = y = u \). Hence,

\[
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
e & \varepsilon & \varepsilon \\
e & \varepsilon & \varepsilon \\
\varepsilon & \varepsilon & \varepsilon
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}.
\]

(6.5)
Block diagram representation of the model is shown in Figure 6.3(b)

### 6.6 Single machine processing a part: there is an infinite random access buffer in front of the machine

A machine preceded by an unlimited buffer can be represented by the block diagram shown in Figure 6.4(a), which can be reduced to the block diagram shown in Figure 6.4(b). Then, the system is described by the following equation

\[
\begin{bmatrix}
  x \\
  y \\
  z \\
\end{bmatrix} =
\begin{bmatrix}
  t & e & e \\
  e & e & e \\
  e & e & e \\
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  w \\
\end{bmatrix},
\]

(6.6)

![Diagram of machine with infinite buffer](image)

Figure 6.4: Machine with an infinite buffer in front of it processing a part
7 Numerical Example

The model is applied to a job shop numerical example. The example introduced in this chapter is also used later in this work.

7.1 Problem Statement

Consider a job shop system with 3 machines, $m_1$, $m_2$ and $m_3$, and 3 parts, $n_1$, $n_2$ and $n_3$. The order in which $n_i$, where $i \in \{1, 2, 3\}$, is processed on machines is given by $m_{n_i}$. The processing order of parts on $m_i$, where $i \in \{1, 2, 3\}$, is given by $n_{m_i}$. In this example we have

$$n_{m_1} = \begin{bmatrix} n_1 & n_2 \end{bmatrix}, \quad n_{m_2} = \begin{bmatrix} n_1 & n_3 \end{bmatrix}, \quad n_{m_3} = \begin{bmatrix} n_3 & n_2 \end{bmatrix},$$

$$m_{n_1} = \begin{bmatrix} m_1 & m_2 \end{bmatrix}, \quad m_{n_2} = \begin{bmatrix} m_1 & m_3 \end{bmatrix}, \quad m_{n_3} = \begin{bmatrix} m_2 & m_3 \end{bmatrix}.$$

The configuration of the job shop is graphically illustrated in a routing diagram in Figure 7.1. In the diagram, horizontal connections describe routes of parts through the machines. For example, it can be seen that part $n_1$ is first processed by $m_1$ and then by $m_2$ and therefore $m_{n_1} = \begin{bmatrix} m_1 & m_2 \end{bmatrix}$. Vertical connections describe order in which parts are processed on machines. For example, it can be seen that order of parts on machine $m_2$ is $n_1$ followed by $n_3$, hence $n_{m_2} = \begin{bmatrix} n_1 & n_3 \end{bmatrix}$. Each node $S_i$ for $i \in \{1, 2, \ldots, 6\}$ represents an operation. For example $S_3$ models an operation corresponding to $m_2$ processing $n_1$. It is assumed that the machines are never blocked, i.e. there is sufficient buffer storage in front of every machine. Therefore $S_i$ is modeled by (6.6), i.e.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} t_i & \varepsilon & t_i \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix},$$

where $t_i$ is processing time for operation $S_i$, where $t_1 = 3, t_2 = 2, t_3 = 2, t_4 = 3, t_5 = 2$, and $t_6 = 1$. Detailed block diagram of the system is shown in Figure 7.2.
We have

\[ S_1 = (F_1, m_1, n_1, n_1), \quad S_2 = (F_2, m_2, n_1, n_1), \]
\[ S_3 = (F_3, m_1, n_2, n_2), \quad S_4 = (F_4, m_3, n_2, n_2), \]
\[ S_5 = (F_5, m_2, n_3, n_3), \quad S_6 = (F_6, m_3, n_3, n_3). \]
It is assumed that the system is not blocked from the outside, i.e. $v_c = x_c$. The whole system, i.e., the job shop, is modeled by

$$S_c = \left( F^{nb}_c, m_c, n^{in}_c, n^{out}_c \right) = \begin{pmatrix} m_1 & n_1 & n_1 \\ m_2 & n_2 & n_2 \\ m_3 & n_3 & n_3 \end{pmatrix},$$

where $F^{nb}$ ($nb$ in $F^{nb}$ stands for “non-blocking system”) describes the following relation

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = F^{nb}_c \begin{bmatrix} u_c \\ w_c \end{bmatrix}.$$  

The goal is to find $F^{nb}$.

### 7.2 Job Shop Model

Most of the matrices derived in this section are sparse, i.e. they contain many elements that equal $\varepsilon$. Therefore, for simplicity, $\varepsilon$’s in all the matrices presented in this Section are replaced by dots.

There is a network of manufacturing blocks $S_1, S_2, \ldots, S_6$ that are composed in one step to produce $S_c$. We have

$$\mathbf{m} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \quad \mathbf{n}^{in} = \mathbf{n}^{out} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}, \quad \mathbf{m}_c = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}, \quad \mathbf{n}^{out}_c = \mathbf{n}^{in}_c = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}.$$
Part-flow interconnection matrices are given by:

\[
R_{in} = \begin{bmatrix}
  e & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix},
\quad
R = \begin{bmatrix}
  . & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix},
\quad
R_{out} = \begin{bmatrix}
  . & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix}
\]

Resource-flow interconnection matrices are

\[
Q_{in} = \begin{bmatrix}
  e & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix},
\quad
Q = \begin{bmatrix}
  . & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix},
\quad
Q_{out} = \begin{bmatrix}
  . & . & . & . & . \\
  . & e & . & . & . \\
  . & . & e & . & . \\
  . & . & . & e & . \\
  . & . & . & . & e
\end{bmatrix}
\]

We have

\[
\tilde{F} = \begin{bmatrix}
  \tilde{F}_{xu} & \tilde{F}_{xv} & \tilde{F}_{xw} \\
  \tilde{F}_{yu} & \tilde{F}_{yv} & \tilde{F}_{yw} \\
  \tilde{F}_{zu} & \tilde{F}_{zv} & \tilde{F}_{zw}
\end{bmatrix} = \begin{bmatrix}
  P & P \\
  E & . \\
  E & .
\end{bmatrix},
\]

where

\[
P = \begin{bmatrix}
  t_1 & . & . & . & . \\
  . & t_2 & . & . & . \\
  . & . & t_3 & . & . \\
  . & . & . & t_4 & . \\
  . & . & . & . & t_5 \\
  . & . & . & . & . & t_6
\end{bmatrix} = \begin{bmatrix}
  3 & . & . & . & . \\
  . & 2 & . & . & . \\
  . & . & 2 & . & . \\
  . & . & . & 3 & . \\
  . & . & . & . & 2 \\
  . & . & . & . & 1
\end{bmatrix}
\]
Assume that the system is not blocked from the outside, i.e. \( v_c = x_c \). Plugging in values into (5.13) we obtain

\[
\begin{bmatrix}
5 & . & . \\
11 & 5 & 6 \\
8 & . & 3
\end{bmatrix}
\begin{bmatrix}
x_c \\
y_c \\
z_c
\end{bmatrix}
= 
\begin{bmatrix}
5 & 2 & . \\
11 & 8 & 4 \\
8 & 5 & 1
\end{bmatrix}
\begin{bmatrix}
u_c \\
w_c
\end{bmatrix}
\]

Assume \( u_c = [e \ e \ e]^T \) and \( w_c = [e \ e \ e]^T \). Then it follows that

\[
\begin{bmatrix}
5 \\
11 \\
8
\end{bmatrix}
\]

Makespan of the system is the latest time when a part leaves the system and it is equal to 11 time units.
8 **Analytical Tools**

This Chapter describes some of the analytical tools provided by the model. Various methods to model the system are introduced, namely part-based and machine-based modeling approaches. In addition, it is demonstrated how the model can be used for (a) deadlock detection, (b) performance evaluation, (c) scheduling, (d) modeling of cyclically scheduled systems, and (e) studying structural properties of the systems.

8.1 **Deadlock Detection**

Consider system $S_c$, modeled by matrix $F_c$, which is defined with respect to $n_{\text{in}}^c$, $n_{\text{out}}^c$, and $m_c$. If the system has deadlocks, then some jobs that enter the system will never be able to leave it. Suppose that job $n = [n_{\text{out}}^c]_i$ is in deadlock and cannot leave the system, then $[x_i] = +\infty$. This means that $F_c$ contains elements that are equal to $+\infty$; in other words $F_c$ will not exist. On the contrary, if $F_c$ exists then the system is free of deadlocks.

Suppose that $S_c$ is a network of subsystems $S_i$, for $i = 1, 2, \ldots, M$, such that each $S_i$ is deadlock free. Then from (5.10) it follows that $F_c$ exists (and, therefore, $S_c$ is deadlock free) if and only if

$$
\begin{pmatrix}
R & \varepsilon & \varepsilon \\
\varepsilon & R^T & \varepsilon \\
\varepsilon & \varepsilon & Q
\end{pmatrix}
\left(8.1\right)
$$

exists. Multiplication of two square $k$ by $k$ matrices has complexity $O(k^3)$. The Kleen star operation on a $k$ by $k$ matrix can also be computed in at most $O(k^3)$ time [20]. Therefore the complexity of checking the system for deadlocks using (8.1) is $O(k^3)$, where $k = |m_{\text{in}}^c| + |n_{\text{in}}^c| + |n_{\text{out}}^c|$ is the dimension of the matrix in (8.1).
8.2 Modeling Approaches

Consider again the example introduced in Chapter 7. In the example $S_c$ was obtained by composing basic blocks $S_1$ through $S_6$ in one step as shown in Figure 7.1. There are, however, alternative methods to obtain $S_c$, such as *machine-based* and *part-based* modeling methods. These methods are based on dividing the system into components and then composing these components to obtain $S_c$. The part-based modeling approach is illustrated in Figure 8.1. The approach consists of two steps. In the first step the models of each part are derived, i.e. the models of $S_{n1}$, $S_{n2}$ and $S_{n3}$ as shown in Figure 8.1(a). Then, in the second step, $S_c$ is obtained by "vertically" composing $S_{n1}$, $S_{n2}$ and $S_{n3}$ as illustrated in Figure 8.1(b). The machine-based modeling approach is illustrated in Figure 8.2. In the approach, we first obtain models of each machine, $S_{m1}$, $S_{m2}$ and $S_{m3}$, as shown in Figure 8.2(a). Then $S_{m1}$, $S_{m2}$ and $S_{m3}$ are composed "horizontally" to produce $S_c$ as shown in Figure 8.2(b).

![Diagram](image_url)

(a) Step I. Obtain models for each part, $S_{n1}$, $S_{n2}$, and $S_{n3}$.

(b) Step II. Combine $S_{n1}$, $S_{n2}$, and $S_{n3}$ into $S_c$.

Figure 8.1: Part-based approach
8.3 Cyclic Systems

In cyclic (periodic) manufacturing system, an identical set of parts is repeatedly produced. It is assumed that processing times, order of parts on machines, routing of parts through the machines - all stay the same in every cycle. Let $k$ be a cycle index. Timing behavior of a manufacturing system during $k$-th production cycle can be described by the equation of the form

$$
\begin{bmatrix}
x(k) \\
y(k) \\
z(k)
\end{bmatrix} =
\begin{bmatrix}
F_{xx} & F_{xy} & F_{xz} \\
F_{yx} & F_{yy} & F_{yz} \\
F_{zx} & F_{zy} & F_{zz}
\end{bmatrix}
\begin{bmatrix}
u(k) \\
v(k) \\
w(k)
\end{bmatrix},
$$

It is assumed that when $k = 0$ the inputs of the model represent the initial state of the system. During the $k$-th production cycle the system produces the $k$-th set of parts. After the system’s resources are set free by the current set of parts, they immediately become available for the next set of parts, in other words, the resources are “recycled”. Hence, the periodic schedule is characterized by $w(k) = z(k - 1)$ as shown Figure 8.3. Then (8.2) can
be written as
\[
\begin{bmatrix}
x(k)
y(k)
z(k)
\end{bmatrix} =
\begin{bmatrix}
F_{xu} & F_{xv} & u(k) & F_{xw} \\
F_{yu} & F_{yw} & v(k) & F_{yw} \\
F_{zu} & F_{zw} & z(k) & F_{zw}
\end{bmatrix} z(k-1).
\] (8.3)

If one wishes to study periodic behavior of the system and its performance measures it is sufficient to study transient and steady state characteristics of the following iterative equation
\[
z(k) =
\begin{bmatrix}
F_{zu} & F_{cv} & u(k) \\
F_{zu} & F_{cv} & v(k)
\end{bmatrix} 
\oplus 
F_{zv} z(k-1).
\] (8.4)

The block diagram representing cyclically scheduled manufacturing systems is shown in Figure 8.3.

![Figure 8.3: Block representation of a cyclic system](image)

Suppose that the system is autonomous, that is, parts are always available to the system and finished parts are removed from the system as soon as they are ready to leave. Therefore \( u(k) = \varepsilon \) and \( v(k) = x(k) \). Then we have
\[
\begin{align*}
v(k) &= x(k) = F_{xu} u(k) \oplus F_{xv} v(k) \oplus F_{xw} z(k-1) = F_{xv} v(k) \oplus F_{xw} z(k-1) \\
&= (F_{xv})^* F_{xw} z(k-1)
\end{align*}
\]
\[
\begin{align*}
z(k) &= F_{zu} u(k) \oplus F_{cv} v(k) \oplus F_{zw} z(k-1) = F_{cv} v(k) \oplus F_{zw} z(k-1) \\
&= (F_{cv} (F_{xv})^* F_{xw} \oplus F_{zw}) z(k-1).
\end{align*}
\]
Let $F_{auto} = (F_{xz}(F_{xy})^{*}F_{xw} \oplus F_{zw})$ then

$$z(k) = F_{auto}z(k - 1).$$ (8.5)

The dynamic equation of the form (8.5) is perhaps one of the most studied equations in max-plus algebra. Transient behavior of (8.5) was studied in [23, Chapters 3,4]. After some number of iterations the system will achieve a periodic steady state regime. It is shown in Baccelli et al. [3, Chapter 3] that the steady-state period of the system described by (8.5) equals to the eigenvalue of $F_{auto}$ (if it exists), whereas an eigenvector of $F_{auto}$ provides data from which complete periodic scheduling regime of the system can be obtained.

8.4 Performance Evaluation

The use of the model for performance evaluation is fairly straightforward. Note that all equations presented in this subsection are written in regular (not max-plus) algebra.

Let $S$ be a model defined by (5.1) with respect to $m$, $n_{in}$ and $n_{out}$. Consider a part $n = [n_{in}]_i = [n_{out}]_j$. The total amount of time that $n$ spends in the system is given by

$$[v]_j - [y]_i.$$

The total amount of time that the resource $[m]_i$ is used by the system is given by

$$[z]_i - [w]_i.$$

Makespan of the system, $\mu$, is defined as the time span between the time when the first part enters the system and the time when the last part is ready to leave the system, therefore

$$\mu = \max([x]_1, [x]_2, \ldots) - \min([y]_1, [y]_2, \ldots).$$

Calculation of throughput for autonomous cyclic systems was addressed in Section 8.3.
Since the model is based on block-diagram representation, we can also use topological methods for performance evaluation in addition to purely mathematical methods as described in Chapter 10.

8.5 Structural Information

Consider a system modeled by a block diagram. Routing of parts through the system is graphically represented by part-flow block interconnections. Part-flow interconnections allow for tracing the flow of parts through the system. They are defined mathematically by part-flow interconnection matrices $R_{in}$, $R$ and $R_{out}$. Similarly, routing of resources through the system is graphically represented by resource-flow block interconnections. Resource-flow interconnections can be used to trace the flow of resources through the system. They are mathematically defined by resource-flow interconnection matrices $Q_{in}$, $Q$ and $Q_{out}$.

8.6 Application to Scheduling

The scheduling problem for a manufacturing system is usually formulated as follows: find the order in which parts are processed on the machines that optimizes some performance measure of the systems, such as makespan, cycle time or work in process. Order of parts on the machines is given by $Q_{in}$, $Q$ and $Q_{out}$.

Consider a system which is represented as a block diagram, where each block is one of the basic blocks described in Chapter 6. The model can then be reduced to a single block using (5.10), which is a function of $Q_{in}$, $Q$ and $Q_{out}$. Suppose that the performance measure to be optimized in the scheduling problem is the makespan. Makespan of the systems can be computed from (5.10) and it is also a function of $Q_{in}$, $Q$ and $Q_{out}$. Hence the scheduling problem can then be formulated as an optimization problem in which $Q_{in}$, $Q$ and $Q_{out}$ are variables and everything else is fixed.
9 Line Applications

Line applications include models of a single part processed by a set of \( M \) resources and models of a single resource processing a set of \( N \) parts. They are called line applications because in the former case basic blocks are stacked horizontally and in the latter case basic blocks are stacked vertically.

9.1 One Part and \( M \) Resources

Consider a part \( n \) which is processed by a set of resources \( \mathbf{m} = [m_1, m_2, \ldots m_M] \) in the order specified by \( \mathbf{m} \). The model of the system is denoted by \( S_c \) with inputs and outputs \( u_c, v_c, w_c, x_c, y_c, z_c \) that are defined with respect to \( m_c = \mathbf{m}, n_c^{in} = n_c^{out} = n \).

The system \( S_c \) can be modeled by a sequence of \( M \) blocks \( S_{i \in M} \) stacked horizontally as shown in Figure 9.1(a), where each \( S_i \) is a basic block described by

\[
\begin{bmatrix}
x_i \\
y_i \\
z_i
\end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix},
\]

(9.1)

where \( \mathbf{F}_i \) is a system matrix defined by either (6.1), (6.5) or (6.6) depending on the characteristics of the resource \( m_i \).

There are two assumptions: (a) the part is removed from the system as soon as it is ready to leave, therefore \( v_c = x_c \), and (b) the part is available to the system at any time, therefore \( u_c = \varepsilon \). The goal is to find \( \mathbf{F}_{c,zw} \) in

\[
\mathbf{z}_c = \mathbf{F}_{c,zw} \mathbf{w}_c.
\]

Refer to Figure 9.1(b).
The blocks in the diagram are interconnected only through part-flow interconnections. We have $Q_{in} = Q_{out} = E$. We have

$$Q_{in} = E, \quad Q = \varepsilon, \quad Q_{out} = E, \quad (9.2)$$

$$R_{in} = J, \quad R = H, \quad R_{out} = G, \quad \text{where} \quad (9.3)$$

$$J = \begin{bmatrix} e \\ e \\ \vdots \\ e \end{bmatrix}, \quad H = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \vdots \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \quad G = \begin{bmatrix} \varepsilon & \cdots & \varepsilon & \varepsilon \end{bmatrix}.$$  

Substituting (9.2) and (9.3) into (5.4) and (5.5) and using the identity

$$v_c = x_c = G\hat{x}$$
we obtain
\[ \hat{u} = H\tilde{x}, \]
\[ \hat{v} = H^T\tilde{y} \oplus G^Tv_c = H^T\tilde{y} \oplus G^TG\tilde{x}, \] (9.4)
\[ \hat{w} = w_c. \]

and
\[ z_c = \tilde{z}. \] (9.5)

The model of \( S_i \) depends on what \( m_i \) is: \( m_i \) can be a unit capacity machine, a buffer or it can be a machine having an infinite random access buffer in front of it. The following develops model for \( S_c \) when:

- each \( m_i \in M \) is a unit capacity machine that has unlimited random access buffer in front;
- each \( m_i \in M \) is a unit capacity machine or buffer.

### 9.1.1 Each \( m_i \) is a machine preceded by an infinite random access buffer

Suppose that each \( m_i \in M \) is a unit capacity machine preceded by an infinite random access buffer. Then \( S_i \) is modeled by an equation of the form (6.6), i.e.

\[
\begin{bmatrix}
  x_i \\
  y_i \\
  z_i
\end{bmatrix}
= \begin{bmatrix}
  I_i & \varepsilon & I_i \\
  e & \varepsilon & \varepsilon \\
  \varepsilon & e & \varepsilon
\end{bmatrix}
\begin{bmatrix}
  u_i \\
  v_i \\
  w_i
\end{bmatrix}
= F_i
\begin{bmatrix}
  u_i \\
  v_i \\
  w_i
\end{bmatrix},
\] (9.6)

where \( t_i \) is the processing time for the part on \( m_i \). Refer to Figure 9.2(a).

From (9.6) and (5.3) we can write
\[
\begin{bmatrix}
  \tilde{x} \\
  \tilde{y} \\
  \tilde{z}
\end{bmatrix}
= \begin{bmatrix}
  P & \varepsilon & P \\
  E & \varepsilon & \varepsilon \\
  \varepsilon & E & \varepsilon
\end{bmatrix}
\begin{bmatrix}
  \hat{u} \\
  \hat{v} \\
  \hat{w}
\end{bmatrix}
Figure 9.2: Model of a part processed by a set of machines: each \( m_i \) is a machine preceded by an infinite buffer.

where

\[
P = \begin{bmatrix}
  t_1 & \varepsilon & \varepsilon \\
  \varepsilon & t_2 & \varepsilon \\
  \varepsilon & \varepsilon & t_3 \\
  \cdots & \cdots & \cdots \\
  \varepsilon & \varepsilon & t_M
\end{bmatrix}.
\]

Then

\[
\tilde{x} = P\tilde{u} \oplus P\tilde{w} \\
\tilde{y} = \tilde{u} \\
\tilde{z} = \tilde{v}.
\]

(9.7)

From (9.4), (9.5) and (9.7) it follows

\[
\tilde{x} = P\tilde{u} \oplus P\tilde{w} = PH\tilde{x} \oplus Pw_c = (PH)^*Pw_c
\]

\[
z_c = \tilde{z} = \tilde{v} = H^T\tilde{y} \oplus G^T G\tilde{x} = H^T H\tilde{x} \oplus G^T G\tilde{x} = (H^T H \oplus G^T G)\tilde{x}
\]

\[
= \tilde{x} = (PH)^*Pw_c,
\]

where we used the identity \( H^T H \oplus G^T G = E \), which follows from the definitions of \( H \) and \( G \). Therefore

\[
z_c = (PH)^*Pw_c. \quad (9.8)
\]

Refer to Figure 9.2(b).
9.1.2 Each $m_i$ is a unit capacity machine or buffer

Suppose that each $m_{i\in M}$ is a unit capacity machine or buffer. Then $S_i$ is modeled by an equation of the form (6.1), i.e.

$$
\begin{bmatrix}
x_i \\
y_i \\
z_i
\end{bmatrix} =
\begin{bmatrix}
t_i & e & e \\
e & e & e \\
e & e & e
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i \\
w_i
\end{bmatrix} =
\begin{bmatrix}
F_i \\
F_i \\
F_i
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i \\
w_i
\end{bmatrix},
$$

(9.9)

where $t_i$ is the processing time for the part on $m_i$. Refer to Figure 9.3(a).

![Detailed block diagram](image)

(a) Detailed block diagram

![Reduced block diagram](image)

(b) Reduced block diagram of $F_{czw}$

Figure 9.3: Model of a part processed by a set of machines: each $m_i$ is a unit capacity machine or buffer.
From (9.9) and (5.3) we can write

\[
\begin{bmatrix}
\tilde{x} \\
\tilde{y} \\
\tilde{z}
\end{bmatrix} = \begin{bmatrix} P & \varepsilon & P \\
\varepsilon & E & \varepsilon \\
\varepsilon & E & \varepsilon
\end{bmatrix} \begin{bmatrix}
\tilde{u} \\
\tilde{v} \\
\tilde{w}
\end{bmatrix},
\]

where

\[
P = \begin{bmatrix}
t_1 & \varepsilon & \varepsilon \\
\varepsilon & t_2 & \varepsilon \\
\vdots \\
\varepsilon & \varepsilon & t_M
\end{bmatrix}.
\]

Then

\[
\begin{align*}
\tilde{x} &= P\tilde{u} \oplus \tilde{P}\tilde{w}, \\
\tilde{y} &= \tilde{u} \oplus \tilde{w}, \\
\tilde{z} &= \tilde{v}.
\end{align*}
\]

(9.10)

From (9.4), (9.5) and (9.10) it follows

\[
\begin{align*}
\tilde{x} &= P\tilde{u} \oplus \tilde{P}\tilde{w} = PH\tilde{x} \oplus \tilde{P}w_c = (PH)^+Pw_c, \\
z_c &= \tilde{z} = \tilde{v} = H^T\tilde{y} \oplus G^T\tilde{G}\tilde{x} = H^T\tilde{u} \oplus H^T\tilde{w} \oplus G^T\tilde{G}\tilde{x} = H^T\tilde{H}\tilde{x} \oplus G^T\tilde{G}\tilde{x} \oplus H^Tw_c \\
&= (H^T\tilde{H} \oplus G^T\tilde{G})\tilde{x} \oplus H^Tw_c = \tilde{x} \oplus H^Tw_c = (PH)^+Pw_c \oplus H^Tw_c.
\end{align*}
\]

Therefore,

\[
z_c = ((PH)^+P \oplus H^T)w_c.
\]

(9.11)

Refer to Figure 9.3(b).

9.2 N Parts and One Resource

Consider a unit capacity machine \(m\) processing a set of parts \(n = [n_1, n_2, \ldots n_N]\) in the order specified by \(n\). Suppose that \(m\) is neither an assembly or disassembly machine, hence \(n = n_{in} = n_{out}\). The model of the system is denoted by \(S_c\) with inputs and outputs \(u_c, v_c, w_c\) and \(x_c, y_c, z_c\) that are defined with respect to \(m_c = m, n_{cin} = n_{cout} = n\).
\( S_c \) can be modeled by a sequence of \( N \) blocks \( S_{i \in N} \) stacked vertically as shown in Figure 9.4(a), where each \( S_i \) is a basic block represented by an equation of the form (6.1), i.e.

\[
\begin{bmatrix}
  x_i \\
  y_i \\
  z_i 
\end{bmatrix} = 
\begin{bmatrix}
  t_i & e & t_i \\
  e & e & e \\
  e & e & e 
\end{bmatrix} 
\begin{bmatrix}
  u_i \\
  v_i \\
  w_i 
\end{bmatrix} = 
\begin{bmatrix}
  t_i \\
  v_i \\
  w_i 
\end{bmatrix},
\]

where \( t_i \) is the processing time of \( n_i \) on \( m \). Refer to Figure 9.4(b).

The blocks in the block diagram are interconnected only through the resource-flow type of interconnections. We have \( w_i = z_{i-1} \) for \( i > 1 \). Therefore the interconnection matrices for the system take the following form

\[
\begin{align*}
Q_{in} &= J, & Q &= H, & Q_{out} &= G, \\
R_{in} &= E, & R &= \mathcal{E}, & R_{out} &= E.
\end{align*}
\]

(9.13) (9.14)

\( J, H \) and \( G \) are defined as in the previous section.

Substituting (9.13) and (9.14) into (5.4) and (5.5) we obtain

\[
\tilde{u} = u_c,
\]

\[
\tilde{v} = v_c,
\]

\[
\tilde{w} = H\tilde{z} \oplus Jw_c.
\]

and

\[
\begin{align*}
x_c &= \tilde{x}, \\
y_c &= \tilde{y}, \\
z_c &= G\tilde{z}.
\end{align*}
\]

(9.15) (9.16)

From (9.12) and (5.3) we can write

\[
\begin{bmatrix}
  \tilde{x} \\
  \tilde{y} \\
  \tilde{z}
\end{bmatrix} = 
\begin{bmatrix}
  P & e & P \\
  E & e & E \\
  e & E & e
\end{bmatrix} 
\begin{bmatrix}
  \tilde{u} \\
  \tilde{v} \\
  \tilde{w}
\end{bmatrix},
\]
where

$$P = \begin{bmatrix} t_1 & \varepsilon & \varepsilon \\ \varepsilon & t_2 & \varepsilon \\ & \ddots & \ddots \\ \varepsilon & \varepsilon & t_M \end{bmatrix}.$$ 

Then

$$\tilde{x} = P\tilde{u} \oplus P\tilde{w},$$
$$\tilde{y} = \tilde{u} \oplus \tilde{w},$$
$$\tilde{z} = \tilde{v}.$$  \hspace{1cm} (9.17)

From (9.15), (9.16) and (9.17) it follows

$$\tilde{z} = \tilde{v} = v_c,$$
$$y_c = \tilde{y} = \tilde{u} \oplus \tilde{w} = u_c \oplus Hz \oplus Jw_c = u_c \oplus Hv_c \oplus Jw_c,$$
$$x_c = Py = Pu_c \oplus PHv_c \oplus PJw_c,$$
$$z_c = G\tilde{z} = Gv_c.$$ 

Therefore,

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} P & PH & PJ \\ E & H & J \\ \varepsilon & G & \varepsilon \end{bmatrix} \begin{bmatrix} u_c \\ v_c \\ w_c \end{bmatrix}. \hspace{1cm} (9.18)$$

Equation (9.18) models unit capacity machine processing a set of parts. Its block diagram representation is provided in Figure 9.4(c).
Figure 9.4: Model of a resource processing a set of parts.
10 Evaluating Gains of Synchronous Matrix Signal Flow Graphs

A signal flow graph (SFG) topologically portrays a set of simultaneous linear algebraic equations. SFGs are weighted directed graphs where the nodes are variables in the equations and weighted directed arcs represent relationships among the variables. SFGs were originally developed by Mason [37]. They are useful in studying complex engineering systems by representing them as interconnection of relatively simple subsystems.

Similarly, a set of simultaneous linear matrix equations can be graphically represented by a matrix signal flow graph (MSFG). In a MSFG nodes are vectors and arc weights are matrices. A rectangular matrix associated with an arc is called the transmittance of the arc. A MSFG is essentially a noncommutative version of a scalar signal flow graph because matrix multiplication is noncommutative [44]. Hereafter a MSFG over regular algebra is referred to as regular MSFG to differentiate it from a synchronous MSFG, which is defined below.

A synchronous SFG is a SFG over max-plus algebra, i.e. a directed graph that topologically portrays a set of simultaneous linear equations in max-plus algebra. Conceptually, synchronous SFGs are similar to regular SFGs, except that in synchronous SFGs regular addition of signals is replaced by max-plus algebraic addition $\oplus$ and regular multiplication of signals is replaced by max-plus algebraic multiplication $\otimes$.

The graph gain from the source node $x_i$ to the sink node $x_j$ is defined by the matrix $T_{j,i}$, which relates $x_i$ to $x_j$ by $x_j = T_{j,i}x_i$. Graphs that have multiple sources and multiple sinks also have multiple graphs gains designated depending upon which source and sink are under consideration.
Pliam [44] have shown that the classical Mason’s gain formula [37] is valid only for SFGs over commutative rings (e.g., regular SFGs). Gains of signal flow graphs over noncommutative ring (e.g. regular MSFGs) can be graphically evaluated using either a Riegle’s formula [46] or topological procedures described in [47].

A theory of synchronous MSFGs has not been studied yet in the literature. The purpose of this chapter is to develop topological methods for evaluating gains of synchronous MSFGs. The theory developed in this chapter is a generalization of the existing topological methods for evaluating gains of MSFGs.

This chapter is organized as follows. In Section 10.1 present preliminary definitions for synchronous MSFGs are presented. Section 10.2 outlines the differences between regular MSFGs and synchronous MSFGs. In Section 10.3 it is shown how the existing theory for regular MSFGs can be extended to synchronous MSFGs. Section 10.4 illustrates an application of synchronous MSFGs to performance evaluation of manufacturing systems.

10.1 Synchronous MSFGs - Preliminary Definitions

A synchronous MSFG is a directed graph in which every node is associated with a vector $x_j$ and every arc is associated with a matrix $A_{i,j}$ such that for every node $x_j$, there corresponds the matrix equation in max-plus algebra $x_j = \bigoplus (A_{j,i}x_i)$, where the summation is over all arcs terminating on $x_j$. A synchronous MSFG topologically portrays a set of simultaneous linear matrix equations in max-plus algebra. For example, the following set of linear matrix equations in max-plus algebra is represented by the
synchronous MSFG in Figure 10.1:

\[ \begin{align*}
  x_2 &= Ax_1 \oplus Gx_4; \\
  x_3 &= Bx_2 \oplus Hx_4; \\
  x_4 &= Cx_3 \oplus Fx_1; \\
  x_5 &= Dx_4.
\end{align*} \tag{10.1} \]

Figure 10.1: Synchronous MSFG representation of (10.1)

An *arc progression* \( p \) from node \( x_1 \) to \( x_k \) in a directed graph is a sequence of arcs that connects a sequence of nodes, i.e. \( p = \{(x_1, x_2)(x_2, x_3) \ldots (x_{k-1}, x_k)\} \) is an arc progression. A *path* is an arc progression in which no node appears more than once. A path in which the terminal node and the initial node are the same node is a loop. An arc from a node to itself is called a *self-loop*. Given a MSFG, a node that has no terminating arcs is referred to as a source node, and a node that has no originating arcs is referred to as a sink node. The rectangular matrix associated with an arc is called the *transmittance of the arc*. Throughout the paper, it is assumed that if in a arc in a graph is not labeled with the weight, then its transmittance equals \( E \) (identity matrix in max-plus algebra of appropriate dimensions).

The *graph gain* or *graph transmission* from the source node \( x_i \) to the sink node \( x_j \) is defined by the matrix \( T_{j,i} \), which relates \( x_i \) to \( x_j \) by \( x_j = T_{j,i}x_i \). For example, for the graph in Figure 10.1, the graph gain from \( x_1 \) to \( x_5 \) is defined by the matrix equation \( x_5 = T_{5,1}x_1 \).
10.2 Regular vs. Synchronous MSFGs

Regular MSFGs belong the the class of SFGs over noncommutative ring [43]. The classic gain formula of Mason applies only to SFGs over commutative rings [44]. Gains of MSFGs can be evaluated using a formula developed by Riegle [46]. In addition, there are three other topological methods for evaluating gains of regular MSFGs [46]: a) by repeated application of basic graph reduction rules, b) by the return loop method, and c) by optimal topological procedure.

The theory of synchronous MSFGs has not been studied before. The gain methods for regular MSFGs cannot be applied to synchronous MSFGs directly. This is because topological methods for evaluating gains of a regular MSFGs require additive inverse operations as well as evaluating matrix inverses, when the graph contains loops. These operations, however, are not defined in the max-plus algebra. Hence, the difference between regular and synchronous MSFGs lies in the treatment of loops as described below. Consider the following equation in regular algebra

\[ x_2 = Ax_2 + Bx_1. \] (10.2)

The solution for \( x_2 \) in terms of \( x_1 \) is

\[ x_2 = (I - A)^{-1}Bx_1, \] (10.3)

where \( I \) is the identity matrix. Now, consider the analog of (10.2) in the max-plus (dioid) algebra:

\[ x_2 = Ax_2 \oplus Bx_1. \] (10.4)

Its solution is

\[ x_2 = A^*Bx_1. \] (10.5)

Hence, \( A^* \) in max-plus algebra is equivalent to \((I - A)^{-1}\) in regular algebra. This fact is used in the next section to develop topological methods for evaluating gains of
synchronous MSFGs. Equation (10.3) describes the removal of a self-loop in the regular MSFGs as shown in Figure 10.2(a). Similarly, equation (10.5) describes the removal of a self-loop in the synchronous MSFGs as shown in Figure 10.2(b).

\[ x_1 \rightarrow B \rightarrow x_2 \]
\[ x_1 \rightarrow (I - A)^{-1}B \rightarrow x_2 \]

(a) Removal of a self-loop in regular MSFG

\[ x_1 \rightarrow B \rightarrow x_2 \]
\[ x_1 \rightarrow (A)^{-1}B \rightarrow x_2 \]

(b) Removal of a self-loop in synchronous MSFG

Figure 10.2: Removal of a self-loop.

10.3 Topological Methods for Evaluating Gains of Synchronous MSFGs

Using the fact that \( A^* \) in max-plus algebra is equivalent to \( (I - A)^{-1} \) in regular algebra, the gain methods for regular MSFGs can be straightforwardly extended to synchronous MSFGs. The following develops extensions of (a) basic block diagram reduction rules and (b) the return loop method to synchronous MSFGs.

10.3.1 Basic Graph Reduction Rules

By the successive application of basic graph reduction rules, a synchronous MSFG can be reduced to a simplified graph from which the desired graph gains can be obtained directly. For example, the synchronous MSFG in Figure 10.3 is a simplified graph that graphically represents (5.1), such that the gains between the inputs and the outputs can be obtained directly from the graph.
The basic graph reduction rules for both synchronous and regular MSFGs are illustrated in Figure 10.4. It should be noted that the expressions for the transmittances of arcs of regular (synchronous) MSFGs in Figure 10.4 are written in regular (max-plus) algebra. The following lists the basic graph reduction rules for synchronous MSFGs.

1. Series reduction is illustrated in Figure 10.4(a).

2. Parallel reduction is illustrated in Figure 10.4(b).

3. Absorption of a node is illustrated in Figure 10.4(c). The node $x_5$ is absorbed.

4. Removal of a self-loop is illustrated in Figure 10.4(d). The self-loop $A_{55}$ at node $x_5$ is removed. Note the difference between removal of a self-loop in regular vs. synchronous MSFG.

Proofs are straightforward and omitted (e.g. see [46] for the case of regular MSFGs and see the previous section for the removal of a self-loop in synchronous MSFGs).
10.3.2 Return Loop Method

The return loop method is an alternative to the repeated application of the basic reduction rules. The original return loop method for regular MSFGs is described in [47].
In this section, an extension of the return loop method to synchronous MSFGs is described. First we introduce some preliminary definitions. Let $p$ be a path from an input node $k$ to an output node $j$.

- The **path product** of $p$ is the product of arc transmittances of $p$ multiplied in reverse order from node $j$ to node $k$.

- A node in a MSFG is said to be **split** when it is replaced by two nodes, a source node and a sink node such that arcs terminating on the original node are made to terminate on the new sink and all the arcs outgoing from the original node are made to originate at the new source [47].

- The **node transmission** $N_i$ of a node $i$ is the graph transmission between the source and the sink which is created by splitting the node $i$.

- The **node transmission** $N^p_i$ of a node $i$ on path $p$ (from $k$ to $j$) is $N_i$ calculated under the condition that all nodes on $p$ between node $i$ and the output node $j$ are split.

- The **node factor** $\hat{N}^p_i$ of a node $i$ on path $p$ is defined as $\hat{N}^p_i = (N^p_i)^*$. The following is the **return loop method** that yields the graph transmission (graph gain) between the source node $k$ and the sink node $j$.

1. Find all of the paths from the source node $k$ to the sink node $j$.

2. The contribution of the gain by a path $p$ is equal to the path product of $p$ interrupted by the node factor of every node on the path $p$. The node factor of node $i$ is inserted between the arc transmittances of the path product touching node $i$.

3. The graph gain $T_{jk}$ is equal to the (max-plus algebraic) sum of the contributions of the gain by each path from $i$ to $j$, where the summation is over all such paths.
The proof of the return loop method for synchronous MSFGs depends entirely on the proof presented by Riegle and Lin [47] for regular MSFGs. The only noteworthy dissimilarity between the methods is as follows. For regular MSFGs the node factor of the node \( i \) on path \( p \) is defined as \((I - N_i^p)^{-1}\) (where \( I \) is the identity matrix in regular algebra), whereas for synchronous MSFGs the node factor of the node \( i \) on \( p \) is defined as \((N_i^p)^*\).

The proof of the return loop method for synchronous MSFGs depends entirely on the proof presented by Riegle and Lin [47] for regular MSFGs. The only noteworthy dissimilarity between the methods is as follows. For regular MSFGs the node factor of the node \( i \) on path \( P \) is defined as \((I - \hat{C}_i)^{-1}\), whereas for synchronous MSFGs the node factor of the node \( i \) on \( P \) is defined as \((\hat{C}_i)^*\).

**Example 10.3.1.** We evaluate the graph gain from node \( x_1 \) to node \( x_5 \) in the synchronous MSFG in Figure 10.1. Applying the return loop method, we have the following.

1. There are two paths from \( x_1 \) to \( x_5 \): \( p_1 = \{x_1, x_2, x_3, x_4, x_5\} \) with path product DCBA, and \( p_2 = \{x_1, x_4, x_5\} \) with path product DF.

2. As shown in Figure 10.5(a), the node factor of node \( x_2 \) on \( p_1 \) equals the graph transmission from \( x_{2a} \) to \( x_{2b} \) when nodes \( x_2, x_3 \) and \( x_4 \) are split. Therefore,

\[
\hat{N}_{21}^p = (\varepsilon)^* = \mathbf{E}.
\]

The node factor of node \( x_3 \) on \( p_1 \) equals the graph transmission from \( x_{3a} \) to \( x_{3b} \) when \( x_3 \) and \( x_4 \) are split as shown in Figure 10.5(b). Therefore,

\[
\hat{N}_{31}^p = (\varepsilon)^* = \mathbf{E}.
\]

Finally, the node factor of node \( x_4 \) on \( p_1 \) (and on \( p_2 \)) equals the graph transmission from \( x_{4a} \) to \( x_{4b} \) when \( x_4 \) is split as shown in Figure 10.5(c). Therefore,

\[
\hat{N}_{41}^p = \hat{N}_{42}^p = (\mathbf{CH} \oplus \mathbf{CBG})^*.
\]
3. Then the graph gain between $x_1$ and $x_5$ is given by

$$T_{5,1} = D(\hat{N}_{4}^{p_1})C(\hat{N}_{3}^{p_1})B(\hat{N}_{2}^{p_1})A \oplus D(\hat{N}_{4}^{p_2})F$$

$$= D(CH \oplus CBG)^\ast CBA$$

$$\oplus D(CH \oplus CBG)^\ast F.$$
be processing time of part \( n_j \in n \) on machine \( m_i \). Each machine \( m_i \) processing \( n \) can be modeled by an equation of the form (9.18), i.e.

\[
\begin{bmatrix}
x_i \\
y_i \\
z_i
\end{bmatrix} =
\begin{bmatrix}
P_i & P_i H & P_i J \\
E & H & J \\
\varepsilon & G & \varepsilon
\end{bmatrix}
\begin{bmatrix}
u_i \\
v_i \\
w_i
\end{bmatrix},
\]

where

\[
P_i =
\begin{bmatrix}
t_{i,1} & \varepsilon & \varepsilon \\
\varepsilon & t_{i,2} & \varepsilon \\
\varepsilon & \varepsilon & t_{i,N}
\end{bmatrix}.
\]

Figure 10.6(a) shows block diagram representation of (10.6) and Figure 10.6(b) shows the equivalent synchronous MSFG representation of (10.6).

Suppose that the parts are processed on the machines in the order \( m_1 \) then \( m_2 \) and then \( m_3 \). After the parts are complete on \( m_3 \) they are immediately removed from the system, i.e. \( v_3 = x_3 \). The permutation flow shop model can be represented by a
synchronous MSFG in Figure 10.7, where the individual signal flow graphs for each $m_i$ are stacked horizontally, such that

\[
x_1 = u_2, \quad x_2 = u_3; \quad v_1 = y_2, \quad v_2 = y_3.
\]

![Figure 10.7: Synchronous MSFG representation of a permutation flow shop with 3 machines](image)

The following illustrates the application of the return loop methods to the graph in Figure 10.7, where the goal is to find the gain between $u_1$ and $x_3$.

1. There is only one directed path from $u_1$ to $x_3$. The path product is $P_3P_2P_1$.

2. [The node factor of node $u_1$] = $E$

   [The node factor of node $y_1$] = $E$

   [The node factor of node $y_2$] = $(P_1H)^*$

   [The node factor of node $y_3$] = $(P_2(P_1H)^*H)^*$

   [The node factor of node $x_3$] = $(P_3(P_2(P_1H)^*H)^*H)^*$

3. The gain expression is then

\[
T = (P_3(P_2(P_1H)^*H)^*H)^*P_3(P_2(P_1H)^*H)^*P_2(P_1H)^*P_1
\]
and $x_3 = Tu_1$.

Suppose that all the machines are initially available, i.e. $w_1 = w_2 = w_3 = \varepsilon$. Then the makespan of the system, $\mu$, is given by $\mu = x_3 = Tu_1$. 
11 Applications of the Model

In this chapter the models of job-shop and flow-shop are derived using the part-based modeling approach (refer to Section 8.2 for explanation of part-based and machine-based modeling approaches). In addition the formulation of scheduling problem is given for both job-shops and flow-shops.

11.1 Job Shop

Consider a job shop with $M$ machines and $N$ parts. Let $m = [m_1, m_2, \ldots, m_M]$ be an ordered set of machines and let $n = [n_1, n_2, \ldots, n_N]$ be an ordered set of parts in the job shop. Let $j \in N$ and $k \in M$. The order in which $n_j \in n$ is routed through the machines is given by an ordered set of machines $m_{n_j}$. The order in which $m_k \in m$ processes parts is given by an ordered set of parts $n_{m_k}$. Let $t_{i,j}$ be processing time of $n_j$ on $[m_{n_j}]_i$, where $i \in [m_{n_j}]$.

There are several assumptions pertaining to the system as stated below.

1. There are no assembly or disassembly operations.

2. Each machine has an unlimited capacity buffer in front of it to store parts, i.e. it is a non-blocking job shop.

3. Parts are available to the system at any time.

4. Parts are removed from the system as soon as they are ready to leave.

The model of a job shop can be obtained using part-based, machine-based or general derivation method. In what follows the model is derived using part-based modeling approach. Note, however that the model can be derived using either of the approaches. It is demonstrated how the scheduling problem can be formulated for the job-shop.
11.1.1 The Model

Suppose that the system is modeled by a manufacturing block $S_c$ with inputs $u_c, v_c$ and $w_c$ and outputs $x_c, y_c$ and $z_c$, which are defined with respect to $n = n^\text{in} = n^\text{out}$ and $m$. Since parts are available to the system at any time, we have $u_c = \varepsilon$. Also, since parts are removed from the system as soon as they are ready to leave, we have $v_c = x_c$. The objective is to model the system through the relation

$$z_c = F_{c,w} w_c.$$

The system is modeled using the part-based approach. First we derive the models of each of the parts $\{n_1, n_2, \ldots, n_j\}$ processed by the job shop system. Consider a part $n_j \in n$, where $j \in N$. Let $S_{n_j}$ denote the model of part $n_j$ processed by a set of machines $m_{n_j}$. Let $u_c, v_c, w_c$ and $x_c, y_c, z_c$ be the inputs and the outputs of $S_{n_j}$, which are defined with respect to $m_{n_j}$. The model of a part processed by a set of machines preceded by infinite random access buffers is represented by (9.8), hence we have

$$z_{n_j} = (P_{n_j} H)^\tau P_{n_j} w_{n_j}, \quad (11.1)$$

where

$$P_{n_j} = \begin{bmatrix}
  t_{1,j} & \varepsilon & \varepsilon \\
  \varepsilon & t_{2,j} & \varepsilon \\
  \varepsilon & \varepsilon & t_{m_{n_j},j}
\end{bmatrix}.$$

Define

$$A_j = (P_{n_j} H)^\tau P_{n_j}$$

then

$$z_{n_j} = A_j w_{n_j}.$$
In the next step $S_{nj}$, for $j \in N$, are composed into one block $S_c$. Interconnection of blocks depends on order in which parts are processed on the machines; it is specified through the resource-flow interconnection matrices $Q_{in}$, $Q$ and $Q_{out}$. We have

$$\tilde{z} = \tilde{A}\tilde{w}$$

From (5.4), it follows

$$\tilde{w} = Q\tilde{z} \oplus Q_{in}w_c = QA\tilde{w} \oplus Q_{in}w_c = (QA)^*Q_{in}w_c.$$  

From (5.5), we have

$$z_c = Q_{out}\tilde{z} = Q_{out}A\tilde{w} = Q_{out}\tilde{A}(QA)^*Q_{in}w_c. \quad (11.2)$$

**11.1.2 Makespan Computation**

The makespan of the job shop, $\mu$, can be computed as

$$\mu = \max([z_c]_1, [z_c]_2, \ldots, [z_c]_M) - \min([w_c]_1, [w_c]_2, \ldots, [w_c]_M).$$

Suppose that $w_c = \begin{bmatrix} e & e & \ldots & e \end{bmatrix}^T$, then

$$\mu = \max([z_c]_1, [z_c]_2, \ldots, [z_c]_M) = \bigoplus_{i=1}^{M}[z_c]_i = Yz_c = YQ_{out}\tilde{A}(QA)^*Q_{in}, \quad (11.3)$$

where $Y = \begin{bmatrix} e & e & \ldots & e \end{bmatrix}$. The makespan is essentially the latest time when the system is “set free”, i.e. the time when the last part leaves the system.

**11.1.3 Scheduling Problem Formulation**

The objective of the job shop scheduling problem is to find an optimal schedule with respect to some performance characteristics. The optimal schedule is specified through order in which parts are processed on the machines. Suppose that the objective is to
minimize the makespan, \( \mu \), and let \( \mathbf{w}_c = \begin{bmatrix} e & e & e & e \end{bmatrix}^T \). Then the goal of the scheduling problem is to minimize

\[
\mu = \mathbf{YQ}_{out} \tilde{\mathbf{A}} (\mathbf{Q} \tilde{\mathbf{A}})^\ast \mathbf{Q}_{in}.
\]  

(11.4)

The decision variables in (11.4) are the resource-flow interconnection matrices \( \mathbf{Q}_{in} \), \( \mathbf{Q} \) and \( \mathbf{Q}_{out} \), because they specify the order in which parts are processed on the machines. Therefore, the job shop scheduling problem can be formulated as follows: find \( \mathbf{Q}_{in} \), \( \mathbf{Q} \) and \( \mathbf{Q}_{out} \) that minimizes the function \( \mathbf{YQ}_{out} \tilde{\mathbf{A}} (\mathbf{Q} \tilde{\mathbf{A}})^\ast \mathbf{Q}_{in} \mathbf{w}_c \). It should be noted that \( \mathbf{Q}_{in} \), \( \mathbf{Q} \) and \( \mathbf{Q}_{out} \) have their own constraints; the matrices need to be valid routing matrices having at most one \( e \) per each row or column with the rest of the elements being equal \( \varepsilon \).

### 11.1.4 Numerical Example

Consider again the job shop system introduced in the numerical example in Chapter 7. In Chapter 7 the model was obtained using the general approach. The goal of this section is to obtain the model of the system using the part based approach as described in the previous subsection.

The system is modeled in two steps. Step 1 - obtain the models for each part. Let \( S_{n_1} \), \( S_{n_2} \) and \( S_{n_3} \) be models of parts \( n_1 \), \( n_2 \) and \( n_3 \), respectively, processed by the job shop. Each \( S_{n_j} \), \( j \in \{1, 2, 3\} \) is modeled by (11.1).

\( S_{n_1} \) is given by

\[
\mathbf{z}_{n_1} = (\mathbf{P}_{n_1} \mathbf{H})^\ast \mathbf{P}_{n_1} \mathbf{w}_{n_1} = \mathbf{A}_1 \mathbf{w}_{n_1},
\]

where \( \mathbf{z}_{n_1} \) and \( \mathbf{w}_{n_1} \) are defined with respect to \( \mathbf{m}_{n_1} \). Therefore

\[
\mathbf{z}_{n_1} = \begin{bmatrix} 3 & \varepsilon \\ \varepsilon & 2 \end{bmatrix} \begin{bmatrix} \varepsilon \\ e \end{bmatrix}^\ast \begin{bmatrix} 3 & \varepsilon \\ \varepsilon & 2 \end{bmatrix} \begin{bmatrix} \varepsilon \\ 5 \end{bmatrix} \mathbf{w}_{n_1} = \begin{bmatrix} 3 & \varepsilon \\ \varepsilon & 2 \end{bmatrix} \mathbf{w}_{n_1}.
\]

\( S_{n_2} \) is given by

\[
\mathbf{z}_{n_2} = (\mathbf{P}_{n_2} \mathbf{H})^\ast \mathbf{P}_{n_2} \mathbf{w}_{n_2} = \mathbf{A}_2 \mathbf{w}_{n_2},
\]
where \( z_{n_2} \) and \( w_{n_2} \) are defined with respect to \( m_{n_2} \). Therefore

\[
z_{n_2} = \begin{bmatrix} 2 & \varepsilon \\ \varepsilon & 3 \end{bmatrix} \begin{bmatrix} \varepsilon & \varepsilon \\ \varepsilon & 3 \end{bmatrix} \begin{bmatrix} 2 \\ \varepsilon \end{bmatrix} \quad w_{n_2} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \begin{bmatrix} \varepsilon \\ 3 \end{bmatrix}.
\]

\( S_{n_3} \) is given by

\[
z_{n_3} = (P_{n_3} H) P_{n_3} w_{n_3} = A_3 w_{n_3},
\]

where \( z_{n_3} \) and \( w_{n_3} \) are defined with respect to \( m_{n_3} \). Therefore

\[
z_{n_3} = \begin{bmatrix} 2 & \varepsilon \\ \varepsilon & 3 \end{bmatrix} \begin{bmatrix} \varepsilon & \varepsilon \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} 2 \end{bmatrix} \quad w_{n_3} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \end{bmatrix} \begin{bmatrix} \varepsilon \end{bmatrix}.
\]

Step 2 - interconnect \( S_{n_1}, S_{n_2} \) and \( S_{n_3} \) vertically as shown in Figure 8.1(b). Routing matrices are

\[
Q_{in} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \quad Q = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \quad Q_{out} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}.
\]

We have

\[
\tilde{A} = \begin{bmatrix} A_1 & \varepsilon & \varepsilon \\ \varepsilon & A_2 & \varepsilon \\ \varepsilon & \varepsilon & A_3 \end{bmatrix} = \begin{bmatrix} 3 & \varepsilon & \varepsilon & \varepsilon \\ 5 & 2 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 3 & 1 \end{bmatrix}.
\]
Assume that $w_e = \begin{bmatrix} e \\ e \end{bmatrix}$. The makespan of the system can be found from (11.3).

\[
\mu = YQ_{out} \tilde{A}(Q\tilde{A})^* Q_{in} w_e
\]

\[
= \begin{bmatrix} e \\ e \\ e \end{bmatrix} \begin{bmatrix} e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \end{bmatrix} \begin{bmatrix} 3 & e & e & e & e \\ 5 & 2 & e & e & e \\ e & e & 2 & e & e \\ e & e & 5 & 3 & e \\ e & e & e & 2 & e \\ e & e & e & 3 & 1 \end{bmatrix}
\]

\[
\otimes \begin{bmatrix} e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \\ e & e & e & e & e & e & e \end{bmatrix} \begin{bmatrix} 3 & e & e & e & e \\ 5 & 2 & e & e & e \\ e & e & 2 & e & e \\ e & e & 5 & 3 & e \\ e & e & e & 2 & e \\ e & e & e & 3 & 1 \end{bmatrix} \begin{bmatrix} e \\ e \end{bmatrix} = 11.
\]

11.2 Cyclic Permutation Flow-Shop

In this section we develop a model of cyclic permutation flow-shop using the proposed modeling approach. It will be shown how the model can be used for performance evaluation and scheduling of cyclic permutation flow-shops. The problem of scheduling of cyclic permutation flow-shops has also been studied by McCormick et al. [38] and by Nambiar and Judd [41].

11.2.1 The Model

Consider a flow shop set up with $M$ machines. It is assumed that there are no buffers between the machines. It is assumed that the production is cyclic, that is, identical sets of
parts are produced repeatedly. The system is producing a set of $N$ parts during each production cycle. Let $\mathbf{m} = [m_1, m_2, \ldots, m_M]$ be an ordered set of machines and $\mathbf{n} = [n_1, n_2, \ldots, n_N]$ be an ordered set of parts. Each $n_i \in \mathbf{n}$ is processed by the machines in the order specified by $\mathbf{m}$. Each machine processes jobs according to $n_1, n_2, \ldots, n_N, n_1, n_2, \ldots$ sequence in a repetitive manner (hence it is a cyclic permutation flows hop). Each job $n_j \in \mathbf{n}$ requires a processing time $t_{i,j}$ on machine $m_i \in \mathbf{m}$.

In cyclic permutation flow-shop problem the usual assumption is that the parts are always available to the system and that the parts are removed from the system as soon as they are ready to leave the system. Therefore, a manufacturing process for part $n_i$ in cycle $k$ can be modeled by the equation of the form (9.11) or, specifically,

$$z_i(k) = (\mathbf{P}_i(\mathbf{H}^t)^\ast \oplus \mathbf{H}^t)\mathbf{w}_i(k), \quad (11.5)$$

where

$$\mathbf{P}_i = \begin{bmatrix} t_{1,i} & \varepsilon & \varepsilon \\ \varepsilon & t_{2,i} & \varepsilon \\ & & \ddots \\ \varepsilon & \varepsilon & t_{M,i} \end{bmatrix}. $$

The complete model of the system can be represented by a block diagram shown in Figure 11.1, where we have

$$\mathbf{w}_i(k) = \begin{cases} z_N(k-1) & \text{for } i = 1, \\ z_{i-1}(k) & \text{for } 1 < i \leq N, \end{cases} \quad (11.6)$$

because all the parts are processed on machines in the same order, that is, $n_1$ is followed by $n_2$ then by $n_3$ etc.
Define $A_i = P_i (H P_i)^* \otimes H^T$. $A_i$ is well defined (refer to (12.18)):

$$A_i = \begin{bmatrix}
    t_{1,i} & e & \cdots & e \\
    t_{1,j} t_{2,i} & t_{2,i} & \cdots & e \\
    \vdots & \vdots & \ddots & \vdots \\
    t_{1,j} t_{2,i} \cdots t_{N-1,i} & t_{2,j} t_{3,i} \cdots t_{N-1,i} & \cdots & e \\
    t_{1,j} t_{2,i} \cdots t_{N,i} & t_{2,j} t_{3,i} \cdots t_{N,i} & \cdots & t_{N,i}
\end{bmatrix}.$$ 

We have

$$z_i(k) = A w_i(k). \quad (11.7)$$

It follows from (11.6) and (11.7) that

$$z_i(k) = \begin{cases}
    A_1 z_N(k - 1) & \text{for } i = 1 \\
    A_i z_{i-1}(k) & \text{for } 1 < i \leq N.
\end{cases} \quad (11.8)$$

Then we can write

$$z_N(k) = A_N A_{N-1} \otimes \ldots \otimes A_1 z_N(k - 1). \quad (11.9)$$
Define

\[ D = A_N A_{N-1} \otimes \ldots \otimes A_1. \]

Then

\[ z_N(k) = Dz_N(k-1) \quad (11.10) \]

### 11.2.2 Scheduling Problem Formulation

Note that \( D \) depends on the order in which parts are processed on machines, which is also called the permutation. In this example it is assumed that \( n = [n_1, n_2, \ldots, n_N] \), however if, lets say, the permutation was \( n = [n_3, n_2, n_4, n_1] \) then we would have \( D = A_4 A_2 A_3 \). So \( D \) is a function of \( n \).

Performance characteristics of the system can be studied by means of the eigenvector and eigenvalue of \( D \). The eigenvalue of \( D \) is the period of the system in the steady-state. The eigenvector of \( D \) provides the steady state schedule of the system.

Suppose that the goal is to find a permutation \( n \) such that the steady-state period of the system is minimized. Let \( \lambda(D) \) denote the eigenvalue of \( D \). Then the optimization problem for cyclic permutation flow shop can be stated as follows: find permutation \( n \) which minimizes \( \lambda(D) \).

In the following Chapter it will be proven that \( D \) is an inverse Monge matrix. This means that the eigenvalue of \( D \) can be computed in \( O(n) \) time and an eigenvector of \( D \) can be computed in \( O(n^2) \) time.
12 PROPERTIES OF INVERSE MONGE MATRICES IN MAX-PLUS ALGEBRA

This chapter explores the properties of inverse Monge matrices in max-plus algebra. An $n$ by $m$ matrix $A$ is called a Monge matrix, if

$$a_{ij} + a_{kl} \leq a_{il} + a_{kj}, \text{ for all } i < k, j < l.$$  \hspace{1cm} (12.1)

Similarly, $A$ is an inverse Monge matrix, if

$$a_{ij} + a_{kl} \geq a_{il} + a_{kj}, \text{ for all } i < k, j < l.$$  \hspace{1cm} (12.2)

Written in max-plus algebra the inverse Monge property becomes

$$a_{ij}a_{kl} \geq a_{il}a_{kj}.$$  

Many problems admit an easy solution if they involve Monge (or inverse Monge) matrices. This property arises in many practical applications, such as transportation, traveling salesman, dynamic programming, economic lot sizing and path problems [7]. For example, the computation of max-plus algebraic eigenvalue of a matrix becomes trivial if the matrix fulfills the inverse Monge property [18]. Hence, the Monge property plays an important role in discrete optimization and computer science [6]. A great survey on Monge matrices and their applications can be found in [7].

In Section 11.2, a cyclic permutation flow shop system is modeled using the approach developed in this work. The eigenvalue of its system matrix, $D$, yields the cyclic period of the system, and the eigenvector of $D$ allows for computing steady-state periodic schedule of the system. In this work it was discovered that $D$ is an inverse Monge matrix. This discovery is what motivated our in-depth study of the properties of inverse Monge matrices.

For general irreducible matrices in max-plus algebra, no better than $O(n^3)$ algorithm is known for computing its max-plus algebraic eigenvalue and eigenvectors [23]. Gavalec
and Plavka [18] have shown that the eigenvalue of an inverse Monge matrix is simply equal to the max-plus algebraic trace of the matrix. In this work we provide an efficient $O(n^2)$ algorithm for computing an eigenvector of an inverse Monge matrix. Moreover, it is shown that the class of inverse Monge matrices is closed under max-plus algebraic multiplication.

This chapter is organized as follows. First, topological properties of inverse Monge matrices are studied using a special type of synchronous signal flow graphs called i/o graphs. Then it is shown that a class of inverse Monge matrices is closed under max-plus algebraic multiplication. Then an efficient $O(n^2)$ algorithm for computing an eigenvector of an inverse Monge matrix is described. Finally, an algebraic proof is provided that the system matrix for permutation flow shops fulfills the inverse Monge property.

### 12.1 I/O Graph Representation of Matrices in Max-Plus Algebra

**Definition 12.1.1.** A weighted directed graph $G = (N, E)$ is called an i/o graph if it has an ordered set of source (input) nodes $i \in N$ and an ordered set of sink (output) nodes $o \in N$ such that $i \cap o = \emptyset$. The graph is denoted by $G = (N, E, i, o)$.

Examples of i/o graphs are given in Figure 12.1.

The set of all paths from $[i]_i$ to $[o]_j$ in $G$ is denoted by $P^G_{i,j}$. Among $P^G_{i,j}$ the path with the maximum weight is called the *primary path* and is denoted by $q^G_{i,j}$. For example, there are two paths from $x_1$ to $x_6$ in $G_1$ in Figure 12.1(a), namely $p_1 = \{x_1, x_6\}$ with $|p_1|_w = 4$ and $p_2 = \{x_1, x_4, x_6\}$ with $|p_2|_w = 3$. The primary path is then $p_1 = \{x_1, x_6\}$.

To any i/o graph $G$ we can associate a matrix $A$, called a gain matrix of $G$. Suppose $|i| = n$ and $|o| = m$. Then $A$ in an $m \times n$ matrix such that $a_{ij} = |q^G_{i,j}|_w$ if $q^G_{i,j}$ exists, or otherwise $a_{ij} = \epsilon$. If $G$ does not contain cycles then all the paths in $G$ have finite weight and therefore $A \in \mathbb{R}^{m \times n}_{\text{max}}$. 
For example, the gain matrices for $G_1$ in Figure 12.1(a) and $G_2$ in Figure 12.1(b) are, respectively:

$$A_1 = \begin{bmatrix} 4 & 6 & 1 \\ 4 & 7 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -2 & 1 \\ 2 & 6 \end{bmatrix}.$$ 

(a) Graph $G_1$, where $i_1 = [x_1 \ x_2 \ x_3]^T$ and $o_1 = [x_6 \ x_7]^T$.

(b) Graph $G_2$, where $i_2 = [x_8 \ x_9]^T$ and $o_2 = [x_{11} \ x_{12}]^T$.

(c) Graph $G_3 = G_1 \times G_2$, where $i_3 = [x_1 \ x_2 \ x_3]^T$ and $o_3 = [x_{11} \ x_{12}]^T$.

Figure 12.1: I/o graphs and their series composition.

**Definition 12.1.2.** Let $G_1$ and $G_2$ be i/o graphs such that $|o_1| = |i_2|$. Then we can merge $G_1$ and $G_2$ into a single graph $G_3$ such that the sink nodes of $G_1$ become the source nodes of $G_2$. $G_3$ is an i/o graph with $i_3 = i_1$ and $o_3 = o_2$. We say that $G_3$ is a series composition of $G_1$ and $G_2$ which is denoted by $G_1 \times G_2$.

An example of two i/o graphs and their series composition is shown in Figure 12.1(c).
Theorem 12.1.3. Let $G_1$ and $G_2$ be $i/o$ graphs such that $|o_1| = |i_2| = k$. Let $A_1$, $A_2$ and $A_3$ be matrices associated with $G_1$, $G_2$ and $G_3 = G_1 \times G_2$, respectively. Then

$$A_3 = A_2 \otimes A_1$$

Proof. Define $n = |i_1|, k = |o_1| = |i_2|$ and $m = |o_2|$. For $k \in n$ and $l \in m$, $[A_3]_{i,k} = [q_{i,j}^{G_1}]_w$, if $q_{k,l}^{G_1}$ exists, otherwise $[A_3]_{i,k} = \epsilon$.

$q_{i,j}^{G_1}$ is composed of the two primary paths – $q_{k,j}^{G_1}$ and $q_{j,l}^{G_2}$, where $j \in \{1, 2, \ldots, k\}$, where $j$ is chosen such that the weight of $q_{i,j}^{G_1}$ if maximal. Therefore

$$[q_{i,j}^{G_1}]_w = \bigoplus_{j=1}^{k} ([q_{k,j}^{G_1}]_w \otimes [q_{j,l}^{G_2}]_w)$$

It follows that

$$[A_3]_{i,k} = \bigoplus_{j=1}^{k} ([A_1]_{i,j} \otimes [A_2]_{j,k}).$$

Therefore, $A_3 = A_2 \otimes A_1$. \qed

For example, the gain matrix of $G_3$ in Figure 12.1(c) is

$$A_3 = A_2 A_1 = \begin{bmatrix} 5 & 8 & 4 \\ 10 & 13 & 9 \end{bmatrix}.$$

### 12.2 Max-Plus Algebraic Multiplication of Inverse Monge Matrices

In this section we prove that the class of inverse Monge matrices is closed under max-plus algebraic multiplication.

**Definition 12.2.1.** We say that paths $p_1 \in P_{i,j}^{G_1}$ and $p_2 \in P_{j,k}^{G_2}$ cross if $i < k$ and $j < l$.

**Definition 12.2.2.** We say that two paths in $i/o$ graph intersect if they have at least one node in common.
Let \( p_1 = a_1 \xrightarrow{p_{11}} z \xrightarrow{p_{12}} a_2 \) and \( p_2 = b_1 \xrightarrow{p_{21}} z \xrightarrow{p_{22}} b_2 \) be intersecting paths such that their common node is \( z \). Then it is easy to see that there exist paths \( p_3 = a_1 \xrightarrow{p_{11}} z \xrightarrow{p_{22}} b_2 \) and \( p_4 = b_1 \xrightarrow{p_{21}} z \xrightarrow{p_{12}} a_2 \). Note that

\[
|p_3| + |p_4| = |p_1| + |p_2| = |p_{11}| + |p_{12}| + |p_{21}| + |p_{22}|.
\]  \tag{12.3}

The following defines a \( \Xi \) i/o graph.

**Definition 12.2.3.** We say that an i/o graph \( G \) is \( \Xi \) if and only if any pair of crossing paths of \( G \) intersect.

\( \Xi \) is the 14th letter of the Greek alphabet (pronounced 'ksi'). It is sometimes written as 'Xi', thus emphasizing that path crossing 'X' implies intersection 'i'. For example, graph \( G_1 \) in Figure 12.1(a) is \( \Xi \) but \( G_2 \) in Figure 12.1(b) is not.

**Theorem 12.2.4.** Let \( G \) be a \( \Xi \) i/o graph. Then its associated matrix \( A \) is an inverse Monge matrix.

**Proof.** Let \( i < k \) and \( j < l \).

Suppose that \( q_{jk}^G \) and \( q_{il}^G \) exist. \( G \) is \( \Xi \), therefore \( q_{jk}^G \) and \( q_{il}^G \) intersect. From (12.3) it follows that there exist paths \( p_1 \in P_{jk}^G \) and \( p_2 \in P_{lk}^G \) such that

\[
|p_1|_w + |p_2|_w = |q_{jk}^G|_w + |q_{il}^G|_w = a_{kj} + a_{il}.
\]

Since \( |q_{jk}^G|_w = a_{ij} \geq |p_1|_w \) and \( |q_{il}^G|_w = a_{kl} \geq |p_2|_w \),

\[
a_{ij} + a_{kl} \geq a_{kj} + a_{il}
\]

Next, suppose that either \( q_{jk}^G \) or \( q_{il}^G \) does not exist, i.e. \( a_{kj} = \varepsilon \) or \( a_{il} = \varepsilon \). Then

\[
a_{kj} + a_{il} = \varepsilon,
\]

and

\[
a_{ij} + a_{kl} \geq a_{kj} + a_{il}.
\]

Therefore \( A \) is an inverse Monge matrix. \( \square \)
Theorem 12.2.5. For any inverse Monge matrix $A$ we can construct a $\Xi$ i/o graph.

Proof. Consider an inverse Monge matrix $A \in \mathbb{R}^{n \times m}_{\text{max}}$. Let $1 \leq i \leq k \leq n$ and $1 \leq j \leq l \leq m$ and suppose that $a_{il} \neq \varepsilon$ and $a_{kj} \neq \varepsilon$. Since $A$ is inverse Monge, we have

$$a_{i,j} + a_{k,l} \geq a_{i,l} + a_{k,j}. \quad (12.4)$$

We can construct an $\Xi$ graph with source nodes $j$ and $l$ and sink nodes $i$ and $k$. One such possible $\Xi$ graph $G$ is shown in Figure 12.2. Indeed

$$|q_{jl}^{G_l}|_w = a_{il} + 0 = a_{il} \quad (12.6)$$

$$|q_{ij}^{G_j}|_w = a_{ij} + a_{kj} - a_{ij} = a_{kj} \quad (12.7)$$

$$|q_{kl}^{G_k}|_w = \max(a_{kl}, a_{il} + a_{kj} - a_{ij}) = a_{kl}, \quad (12.8)$$

where (12.8) follows from (12.4).

![Figure 12.2: A $\Xi$ i/o graph](image)

The next theorem states that the series composition of two $\Xi$ i/o graphs is again a $\Xi$ i/o graph.

Theorem 12.2.6. If $G_1$ and $C_2$ are both $\Xi$ i/o graphs such that $|o_1| = |i_2|$ then $G = G_1 \times G_2$ is again a $\Xi$ i/o graph.
Proof. Let
\[ p_1 = l \xrightarrow{p_1^1} u \xrightarrow{p_2^1} i \in P_{li}^G, \]
\[ p_2 = j \xrightarrow{p_1^2} v \xrightarrow{p_2^2} k \in P_{jk}^G, \]
denote two crossing paths in \( G \), where \( i < k \), \( j < l \), \( p_1^1 \) and \( p_1^2 \) denote sub-paths of \( p_1 \) belonging to \( G_1 \) and \( G_2 \), respectively and, similarly, \( p_2^1 \) and \( p_2^2 \) denote sub-paths of \( p_2 \) belonging to \( G_1 \) and \( G_2 \), respectively.

If \( u = v \), then \( p_1 \) intersects with \( p_2 \).

If \( u < v \), then \( p_1^1 = l \xrightarrow{p_1^1} u \) and \( p_2^1 = j \xrightarrow{p_2^1} v \) cross. Since \( G_1 \) is a \( \Xi \) i/o graph, \( p_1^1 \) and \( p_2^1 \) must intersect, which implies that \( p_1 \) and \( p_2 \) intersect.

If \( u > v \), then \( p_1^2 = u \xrightarrow{p_1^2} i \) and \( p_2^2 = v \xrightarrow{p_2^2} k \) cross. Since \( G_2 \) is a \( \Xi \) i/o graph, \( p_1^2 \) and \( p_2^2 \) must intersect, which implies that \( p_1 \) and \( p_2 \) intersect.

Hence, \( p_1 \) and \( p_2 \) intersect and, therefore, \( G \) is a \( \Xi \) i/o graph. □

The following theorem states that the class of inverse Monge matrices is closed under max-plus algebraic multiplication.

**Theorem 12.2.7.** If \( A \in \mathbb{R}^{l \times n}_{\max} \) and \( B \in \mathbb{R}^{n \times k}_{\max} \) are matrices fulfilling inverse Monge property then \( AB \) is an inverse Monge matrix.

**Proof.** The proof follows directly from Theorems 12.2.5, 12.2.6, and 12.2.4. □

### 12.3 Computing an Eigenvector of an Inverse Monge Matrix

Let \( B \in \mathbb{R}^{n \times n}_{\max} \) be an irreducible matrix. Let \( [B_{ij}]_{ij} = a_{ij} - \lambda(B) \), where \( \lambda(B) \) is the eigenvalue of \( B \). As stated in Theorem 4.4.2, an eigenvector \( v \) of \( B \) can be computed as follows. Let \( c \) be an integer such that \( [B_+^i]_{c,c} = e \). Then any \( c \)-th column vector of \( B_+^i \) is an eigenvector of both \( B \) and \( B_+^i \). \( B_+^i \) is called the metric matrix of \( B \).

For irreducible matrices of general type an eigenvector of \( B \) can be computed in at most \( O(n^3) \) operations [23]. In this section an efficient \( O(n^3) \) algorithm for computing an
eigenvector of an irreducible inverse Monge matrix $A$ is presented. First, we describe the method for computing the metric matrix of $A$.

### 12.3.1 Computing the Metric Matrix

In this Subsection it is shown that when computing $A_j^+$ it is sufficient to consider only monotonic paths in $H(A)$.

Let $A \in \mathbb{R}^{n \times n}_{\text{max}}$ and let $H(A)$ be its communication graph (refer to Chapter 4 for the definition of communication graph of $A$).

**Definition 12.3.1 ([19]).** Let $p = (i_1, i_2, \ldots, i_r)$ be a path in $H(A)$. We say that a node $i_k \in p$ is a peak in $p$, if

$$i_{k-1} < i_k \quad \text{and} \quad i_k > i_{k+1}.$$  

Similarly, we say that a node $i_k \in p$ is a valley in $p$, if

$$i_{k-1} > i_k \quad \text{and} \quad i_k < i_{k+1}.$$  

A monotonic path is defined through notions of peaks and valleys.

**Definition 12.3.2.** We say that an elementary path or an elementary circuit in $H(A)$ is monotonic if it has no peaks or valleys.

It can be observed that a monotonic path $p = (i_1, i_2, \ldots, i_r)$ in $H(A)$ can be classified into the following three types:

1. $p$ in which all the arcs are increasing, i.e. $i_k < i_{k+1}$, for all $1 \leq k < r$;
2. $p$ in which all the arcs are decreasing, i.e. $i_k > i_{k+1}$, for all $1 \leq k < r$;
3. a monotonic circuit is always just a self-loop, for example $\gamma = (i_1, i_1)$.  


Let $A_\lambda = A - \lambda = A \otimes (-\lambda)$, where $\lambda$ is the eigenvalue of an irreducible matrix $A$. All elementary cycles in $A_\lambda$ are non-positive (see Chapter 4). In what follows it will be shown how the notion of monotonic path can be used for efficient computation of $A_\lambda^+$, when $A \in \mathbb{R}_{\text{max}}^{n \times n}$ is an inverse Monge matrix. A monotonic path is defined dually with respect to a spiral path (spiral) described in [19]. In fact, the role that monotonic paths play for inverse Monge matrices is identical to the role that spirals play for Monge matrices.

**Theorem 12.3.3.** Let $A_\lambda$ be an inverse Monge matrix, in which $\lambda(A_\lambda) = e$. Let $P_{\text{mono}}(i, j)$ denote the set of all monotonic paths in $H(A_\lambda)$ from $i$ to $j$. Then

$$[A_\lambda^+]_{i,j} = \max \{|p|_w : p \in P_{\text{mono}}(j, i)\}.$$ 

**Proof.** According to [20], $[A_\lambda^+]_{i,j}$ equals to the maximal weight among all the elementary paths in $H(A_\lambda)$ from $j$ to $i$. In what follows it will be shown that when $A_\lambda$ is an inverse Monge matrix, only monotonic paths need to be considered when searching for the maximum weights of paths in $H(A_\lambda)$.

Consider an elementary path $p = (i_1, i_2, \ldots, i_r)$ in $H(A_\lambda)$ from $i = i_1$ to $j = i_r$ and suppose it is not monotonic. By Definition 12.3.2, $p$ contains at least one peak or valley. Suppose that $i_k$ is either a peak in $p$ with

$$i_{k-1} < i_k \quad \text{and} \quad i_k > i_{k+1},$$

or a valley in $p$ with

$$i_{k-1} > i_k \quad \text{and} \quad i_k < i_{k+1}.$$

Then, since $A_\lambda$ fulfills the inverse Monge property, we have

$$[A_\lambda]_{i_k,i_{k+1}} + [A_\lambda]_{i_{k+1},i_k} \leq [A_\lambda]_{i_k,i_{k+1}} + [A_\lambda]_{i_{k+1},i_{k-1}}. \quad (12.9)$$
It follows that $[A_i]_{i_k,i_k} \neq \varepsilon$, $[A_{i+1,i+1}]_{i_{k-1},i_{k-1}} \neq \varepsilon$ and so arcs $(i_k, i_k)$ and $(i_{k-1}, i_{k+1})$ in $H(A_i)$ must exist. Therefore from (12.9),

$$|(i_{k-1}, i_k)|_w + |(i_k, i_{k+1})|_w \leq |(i_k, i_k)|_w + |(i_{k-1}, i_{k+1})|_w.$$

(12.10)

The path $p$ can be expressed as

$$p = i_1 \xrightarrow{p_1}(i_{k-1}, i_k), (i_k, i_{k+1}) \xrightarrow{p_2} i_r.$$

If we substitute the arcs $(i_{k-1}, i_k)$ and $(i_k, i_{k+1})$ in $p$ by $(i_k, i_k)$ and $(i_{k-1}, i_{k+1})$ we get a sub-path

$$p' = i_1 \xrightarrow{p_1}(i_{k-1}, i_k) \xrightarrow{p_2} i_r$$

and a cycle

$$c = (i_k, i_k).$$

It follows from (12.10) that $|p'|_w + |c|_w \geq |p|_w$. Since $|c|_w$ is non-positive, we have

$$|p'|_w \geq |p|_w.$$

If $p'$ still has peaks and/or valleys then the above procedure can be applied again, this time to $p'$, and so on, until a monotonic path in $H(A_i)$ is yielded, the weight of which is greater than or equal to $|p|_w$. The number of such iterations is finite because $|p'| = |p| - 1$ and because a path consisting of just one arc is always monotonic.

Hence, when searching for the maximum weight of an elementary path in $H(A_i)$ from $i$ to $j$, the set of paths under consideration can be limited to $p \in P_{mono}(i, j)$.

**Theorem 12.3.4.** Let $A$ be an irreducible matrix with $\lambda(A) = 0$. Suppose that $A$ fulfills the inverse Monge property. Then

$$[A^+]_{i,j} = \begin{cases} [A]_{i,j}, & \text{for } j - 1 \leq i \leq j + 1, \\ [A]_{i,j} \oplus \bigoplus_{k=j+1}^{i-1} ([A^+]_{k,j}[A]_{i,k}), & \text{for } i > j + 1, \\ [A]_{i,j} \oplus \bigoplus_{k=i+1}^{j-1} ([A^+]_{k,j}[A]_{i,k}), & \text{for } i < j - 1. \end{cases}$$

(12.11)
Proof. \([A^+]_{ij}\) is equal to the maximal weight among all the elementary paths in \(H(A_\lambda)\) from \(j\) to \(i\). Theorem 12.3.3 states when computing \([A^+]_{ij}\) it is sufficient to consider only the paths in \(P_{\text{mono}}(A_\lambda; j, i)\).

Let \(j - 1 \leq i \leq j + 1\). There is only one elementary monotonic path from node \(i\) to itself, which is self-loop \((i, i)\). There is only one monotonic path from \(j - 1\) to \(i\), which is arc \((j - 1, i)\). There is only one monotonic path from \(j + 1\) to \(i\), which is arc \((j + 1, i)\). Therefore

\[
|j \rightarrow [A^+]_{ij} \rightarrow i|_w = |j \rightarrow [A]_{ij} \rightarrow i|_w.
\]

and

\[
[A^+]_{i,j} = [A]_{i,j}.
\]

Let \(j + 2 \leq i \leq N\). When computing \(|j \rightarrow [A^+]_{ij} \rightarrow i|_w\) we can first consider the set of monotonically increasing paths containing the arc \((j, i)\) (such set has only the arc \((j, i)\) itself), then the set of set of monotonically increasing paths containing the arc \((j + 1, i)\), then the set of set of monotonically increasing paths containing the arc \((j + 2, i)\), and so forth. Then

\[
|j \rightarrow [A^+]_{ij} \rightarrow i|_w = |j \rightarrow [A]_{ij} \rightarrow i|_w
\]

\[
\oplus |j \rightarrow [A^+]_{j+1, j} \rightarrow j + 1 \rightarrow [A]_{j+1, j} \rightarrow i|_w
\]

\[
\oplus |j \rightarrow [A^+]_{j+2, j} \rightarrow j + 2 \rightarrow [A]_{j+2, j} \rightarrow i|_w
\]

\[
\vdots
\]

\[
\oplus |j \rightarrow [A^+]_{i-1, j} \rightarrow i - 1 \rightarrow [A]_{i-1, j} \rightarrow i|_w.
\]

Therefore

\[
[A^+]_{i,j} = [A]_{i,j} \oplus [A^+]_{j+1, j} [A]_{i,j+1} \oplus [A^+]_{j+2, j} [A]_{i,j+2} \oplus \ldots \oplus [A^+]_{i-1, j} [A]_{i,j-1}
\]

\[
= [A]_{i,j} \oplus \bigoplus_{k=j+1}^{i-1} ([A^+]_{k,j} [A]_{i,k}).
\]
Let $1 \leq i \leq j - 2$. When computing $|j \rightarrow [A^+]_{i,j}|_w$ we can first consider the set of monotonically increasing paths containing the arc $(j, i)$ (such set has only the arc $(j, i)$ itself), then we consider the set of monotonically decreasing paths containing the arc $(i + 1, i)$, then the set of set of monotonically decreasing paths containing the arc $(i + 2, i)$, and so forth. Then

$$|j \rightarrow [A^+]_{i,j}|_w = |j \rightarrow [A]_{i,j}|_w$$

$$\oplus |j \rightarrow [A^+]_{i+1,j}|_w$$

$$\oplus |j \rightarrow [A^+]_{i+2,j}|_w$$

$$\vdots$$

$$\oplus |j \rightarrow [A^+]_{j-1,j}|_w$$

Therefore

$$[A^+]_{i,j} = [A]_{i,j} \oplus [A^+]_{i+1,j} [A]_{i,j+1} \oplus [A^+]_{i+2,j} [A]_{i,j+2} \oplus \ldots \oplus [A^+]_{j-1,j} [A]_{i,j-1}$$

$$= [A]_{i,j} \oplus \bigoplus_{k=i+1}^{j-1} ([A^+]_{k,j} [A]_{i,k}).$$

\[ \square \]

12.3.2 An Algorithm for Computing an Eigenvector

Define $A_\lambda = A - \lambda(A)$, where $\lambda(A) = \bigoplus_{i=1}^n a_{i,i}$ is the eigenvalue of $A$. Let $c$ be an integer such that $[A^+]_{c,c} = e$. Then, according to [20], any $c$-th column vector of $A^+_\lambda$ is an eigenvector of both $A$ and $A_\lambda$, i.e.

$$[v]_i = [A^+]_{i,c}.$$

Theorem 12.3.4 states that $[A^+]_{c,c} = [A_{\lambda}]_{c,c} = e$, therefore $c$ can be also defined as an integer for which $[A_{\lambda}]_{c,c} = e$.

The following presents an algorithm for computing $v$. The algorithm recursively computes the $c$-th column of $A^+_\lambda$ using (12.11) from Theorem 12.3.4.
**Require:** An irreducible inverse Monge matrix $A \in \mathbb{R}_{\text{max}}^{n \times n}$.

**Ensure:** The eigenvalue $\lambda(A)$ and an eigenvector $v$ of $A$.

1: $\lambda(A) = \bigoplus_{i=1}^{n} a_{i,i}$.

2: $A_\lambda = A - \lambda(A)$.

3: Find $c$, such that $[A_\lambda]_{c,c} = e$.

4: $[v]_{c-1} = [A_\lambda]_{c-1,c}$

5: $[v]_c = [A_\lambda]_{c,c}$

6: $[v]_{c+1} = [A_\lambda]_{c+1,c}$

7: for $l = c + 2, c + 3$ to $n$ do

8: $[v]_{l} = [A_\lambda]_{l,c} \oplus \bigoplus_{k=c+1}^{l-1} [v]_k [A_\lambda]_{l,k}$

9: end for

10: for $l = c - 2, c - 3$ to $1$ do

11: $[v]_{l} = [A_\lambda]_{l,c} \oplus \bigoplus_{k=l+1}^{c-1} [v]_k [A_\lambda]_{l,k}$

12: end for

It can be seen that in the worst possible case $[v]_i, i \in \{1, 2, \ldots, n\}$, can be computed using at most $2(n - 1)$ operations. Therefore the complexity of the algorithm is $O(n^2)$.

### 12.4 Monge Property in Modeling of Cyclic Permutation Flow Shops

The system matrix for permutation flow shop is given by the following algebraic expression (see Section 11.2):

$$D = A_N A_{N-1} \otimes \ldots \otimes A_1,$$

where $A_i = P_i (HP_i)^* \oplus H^T$. In this section it is proven that $D$ is an inverse Monge matrix. Since the class of inverse Monge matrices is closed under max-plus algebraic multiplication as proven in the previous section, it is sufficient to show that $A_i$, for all $i \in \mathbb{N}$, is an inverse Monge matrix. Define $A = P(HP)^* \oplus H^T$, where $P, H \in \mathbb{R}_{\text{max}}^{M \times M}$.
are defined as follows

\[
P = \begin{bmatrix}
t_1 & \varepsilon & \varepsilon \\
\varepsilon & t_2 & \varepsilon \\
\vdots & \ddots & \ddots \\
\varepsilon & \varepsilon & t_M 
\end{bmatrix}, \quad H = \begin{bmatrix}
\varepsilon & \ldots & \varepsilon & \varepsilon \\
\varepsilon & \ldots & \varepsilon \\
\vdots & \ddots & \ddots & \ddots \\
\varepsilon & \ldots & \varepsilon & \varepsilon
\end{bmatrix}.
\]

Then the objective is to show that \( A \) fulfills the inverse Monge property.

Define

\[
\delta_{i,j} = \begin{cases}
e & \text{if } i = j; \\
\varepsilon & \text{if } i \neq j
\end{cases}
\]

\[
\tau_{i,j} = \begin{cases}
\bigotimes_{k=j}^{i} t_k & \text{for } 1 \leq i \leq j; \\
\varepsilon & \text{otherwise}
\end{cases}
\]

Then

\[
[P]_{i,j} = t_i \delta_{i,j}
\]

\[
[H]_{i,j} = \delta_{i-1,j}
\]

We have

\[
[HP]_{i,j} = \bigoplus_{k=1}^{M} \delta_{i-1,k} t_k \delta_{k,j} = \delta_{i-1,j} \tau_{i-1,i-1},
\]

because \( \delta_{k,j} = e \neq \varepsilon \) only if \( k = j \).

**Theorem 12.4.1.**

\[
[(HP)^n]_{i,j} = \delta_{i-n,j} \tau_{i-1,i-n}
\]
Proof. Equation (12.12) proves the case for \( n = 1 \). Assume this is true for \( n > 1 \).

\[
[(HP)^{n+1}]_{i,j} = [HP(HP)^n]_{i,j} = \bigoplus_k^M \left( \delta_{i-1,k}t_{i-1} \delta_{k-n,j} \tau_{k-1,k-n} \right)
\]

note that \( \delta_{i-1,k} = e \neq \varepsilon \) only if \( k = i - 1 \), hence

\[
= t_{i-1} \delta_{i-1-n,j} \tau_{i-2,i-1-n}
\]

note that \( t_{i-1} \tau_{i-2,i-1-n} = \tau_{i-1,i-(n+1)} \), hence

\[
= \delta_{i-(n+1),j} \tau_{i-1,i-(n+1)}.
\]

\(\square\)

Then

\[
\left[ \bigoplus_{n=1}^\infty (HP)^n \right]_{i,j} = \bigoplus_{n=1}^\infty \left( \delta_{i-n,j} \tau_{i-1,i-n} \right) = \tau_{i-1,j},
\]

because \( \delta_{i-n,j} = e \neq \varepsilon \) only if \( i - n = j \). We have

\[
(HP)^* = E \oplus \bigoplus_{n=1}^\infty (HP)^n.
\]

Then

\[
[(HP)^*]_{i,j} = \delta_{i,j} \oplus \tau_{i-1,j}.
\]

It follows that

\[
[P(HP)^*]_{i,j} = \bigoplus_{k=1}^M \left( t_i \delta_{i,k} (\delta_{k,j} \oplus \tau_{k-1,j}) \right)
\]

\[
= \bigoplus_{k=1}^M t_i \delta_{i,k} \delta_{k,j} \oplus \bigoplus_{k=1}^M t_i \delta_{i,k} \tau_{k-1,j}
\]

\[
= t_i \delta_{i,j} \oplus \tau_{i,j} = \tau_{i,j}.
\]

Then

\[
[P(HP)^* \oplus H^T]_{i,j} = \tau_{i,j} \oplus \delta_{i+1,j}
\]

\[
= \begin{cases} 
\tau_{i,j} & \text{for } i \leq j \\
\varepsilon & \text{for } i = j + 1 \\
e & \text{for } i > j + 1
\end{cases}
\]
Therefore, $P(HP)^* \oplus H^T$ can be written as follows

$$P(HP)^* \oplus H^T = \begin{bmatrix} 
 t_1 & e & \cdots & e \\
 t_1t_2 & t_2 & \cdots & e \\
 \vdots & \vdots & \ddots & \vdots \\
 t_1t_2 \ldots I_{N-1} & t_2I_3 \ldots I_{N-1} & \cdots & e \\
 t_1t_2 \ldots I_N & t_2I_3 \ldots I_N & \cdots & I_N
\end{bmatrix}.$$

(12.18)

Theorem 12.4.2. $A = P(HP)^* \oplus H^T$ is an inverse Monge matrix.

Proof. Let $i < k$ and $j < l$.

We will explore the inverse Monge property for the three cases, namely for $i \leq l$, $i = l + 1$ and $i > l + 1$.

Suppose that $i \leq l$, then

$$[A]_{i,j}[A]_{k,l} = \tau_{i,j}\tau_{k,l} = \tau_{i,l}\tau_{l,j},$$

because $\tau_{i,j} = \tau_{l,j}\tau_{l,j}$. In addition,

$$[A]_{i,j}[A]_{k,j} = \tau_{i,j}\tau_{k,j} = \tau_{i,l}\tau_{l,j},$$

because $\tau_{k,j} = \tau_{k,j}\tau_{l,j}$. Therefore $[A]_{i,j}[A]_{k,l} = [A]_{i,l}[A]_{k,j}$.

Next, suppose that $i = l + 1$, then

$$[A]_{i,j}[A]_{k,l} = \tau_{i,j}\tau_{k,l} = \tau_{i,l}\tau_{l,l},$$

because $\tau_{i,j} = \tau_{i+1,j} = \tau_{i+1,j}\tau_{l,j} = e\tau_{l,j} = \tau_{l,j}$. In addition,

$$[A]_{i,j}[A]_{k,j} = \tau_{i,j}\tau_{k,j} = \tau_{k,j}\tau_{l,j},$$

because $\tau_{i,j} = \tau_{i+1,j} = e$ and $\tau_{k,j} = \tau_{k,j}\tau_{l,j}$. Therefore $[A]_{i,j}[A]_{k,l} = [A]_{i,l}[A]_{k,j}$.

Finally, suppose that $i > l + 1$, then

$$[A]_{i,j}[A]_{k,l} = e\tau_{k,j} = e.$$
Hence $[A]_{i,j}[A]_{k,l} \geq [A]_{i,l}[A]_{k,j}$.

Equation $[A]_{i,j}[A]_{k,l} \geq [A]_{i,l}[A]_{k,j}$ holds for any $i < k$ and $j < l$ and therefore A is an inverse Monge matrix. □
13 Conclusion

The dissertation have described the new modeling approach for deterministic manufacturing systems. The approach is based on block diagrams. The blocks in the block diagram are interconnected through a) part-flow interconnections, which specify flow of parts through the diagram, and b) resource-flow interconnections, which specify flow of resources through the diagram. The model is hierarchical - it has been shown how a network of blocks can be combined into one block that has the same input output structure. A complex manufacturing system can be built from a set of basic manufacturing blocks. The set of elementary manufacturing blocks consists of 1) model of a machine processing a part, 2) models of assembly and disassembly machines, and 3) models of unit capacity and infinite capacity buffers.

The model has two representations (a) a mathematical one, expressed in terms of a set of simultaneous linear matrix equations in max-plus algebra, and (b) a graphical one, expressed in terms of a synchronous MSFG. From the mathematical perspective, we derived a formula for composing a network of manufacturing blocks into a single manufacturing block. From the synchronous MSFG perspective, we have looked at topological methods for performance evaluation of the model and presented new topological methods for evaluating gains of synchronous MSFGs.

A number of analytical tools provided by the model have been discussed. The model can be used for (a) performance evaluation, (b) scheduling, (c) deadlock detection, (d) structural analysis, (e) design, and (f) control of deterministic manufacturing systems. In addition, the model can be applied to cyclic (periodically scheduled) deterministic manufacturing systems.

An application of the model have been illustrated by two examples. Firstly, it has been shown how the model can be used to formulate job shop scheduling problem. The new formulation of the scheduling problem is believed to allow for developing new and
possibly more effective scheduling heuristics as compared to the existing scheduling heuristics. Secondly, a cyclic permutation flow shop system has been modeled. The eigenvalue of its system matrix, $\mathbf{D}$, yields the cyclic period of the system, and the eigenvector of $\mathbf{D}$ allows for computing steady-state periodic schedule of the system.

In this work it was discovered that $\mathbf{D}$ is an inverse Monge matrix. The discovery motivated our in-depth study of the properties of inverse Monge matrices, which resulted in an efficient way of computing steady-state performance characteristics of cyclic permutation flow shops. The following lists contributions of this work to the theory of inverse Monge matrices over max-plus algebra.

- it was shown that the system matrix of cyclic permutation flow shops fulfils the inverse Monge property;

- an efficient $O(n^2)$ algorithm for computing a max-plus algebraic eigenvector of an inverse Monge matrix $\mathbf{A} \in \mathbb{R}^{n \times n}_{\text{max}}$ was presented;

- the proof that the class of inverse Monge matrices is closed under max-plus algebraic multiplication was provided.
14 Future Work

The following lists main directions for future work.

- **Implementing the model is software.** The implementation should be fairly straightforward because the model essentially involves addition and multiplication of matrices in the max-plus algebra.

- **Developing scheduling heuristics based on the new model.** The new modeling formalism can be used in various scheduling heuristics such as Tabu search or genetic algorithm.

- **Developing hierarchical approach to modeling of DESs.** A common feature of many discrete event systems is the interaction of entities with resources. For example, in a manufacturing system, parts (entities) are processed by machines (resources), as illustrated in Figure 14.1(a). Similarly, in a railroad transportation system, trains (entities) run on railroad tracks (resources), as shown in Figure 14.1(b). Further, in a computing system, tasks (entities) require CPU time (resources), as shown in Figure 14.1(c). Therefore, if the behavior of a DES is deterministic, than the DES can be modeled by a general hierarchical max-plus algebraic modeling approach that is similar in concept to the model developed for manufacturing systems. I intend to develop such a model.

- **Extending the model to stochastic systems.** Currently the new model applies to deterministic manufacturing systems. Heidergott [22] have studied max-plus linear stochastic systems and perturbation analysis. Based on that knowledge, in the future it may be possible to develop an extension of the model to stochastic manufacturing systems.
Figure 14.1: The concept of entities competing for resources.
REFERENCES


