Efficient Algorithms for Calculating the System Matrix and the Kleene Star Operator for Systems Defined by Directed Acyclic Graphs over Dioids

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Esmaeil Bahalkeh

December 2015

This thesis titled

Efficient Algorithms for Calculating the System Matrix and the Kleene Star Operator for Systems Defined by Directed Acyclic Graphs over Dioids

by

ESMAEIL BAHALKEH

has been approved for

the Department of Industrial and Systems Engineering

and the Russ College of Engineering and Technology by

Robert P. Judd

Professor of Industrial and Systems Engineering

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

BAHALKEH, ESMAEIL, M.S., December 2015, Industrial and Systems Engineering

Efficient Algorithms for Calculating the System Matrix and the Kleene Star Operator for Systems Defined by Directed Acyclic Graphs over Dioids

Director of Thesis: Robert P. Judd

Calculating the performance measures of a manufacturing system is of fundamental importance in many industrial engineering problems, particularly scheduling. The max-plus algebra representation of a system is given by the system matrix, shown by **A**, that can be used to calculate different performance measurements.

This study proposes an efficient and structured algorithm to calculate **A** from the graph representation of the system. The proposed algorithm uses Kahn's algorithm [1] to topologically sort the nodes in the graph.

In addition, using the same algorithm the Kleene star of a matrix can be computed in a slightly more efficient way. Kleene star contains longest path values between any two vertices. It is used to solve linear equations and also for performance measure purposes.

Moreover, this study performs interval analysis on proposed algorithms. In interval system, unlike deterministic systems, input data of the system such as processing times are shown by intervals.

ACKNOWLEDGEMENTS

This thesis would not have been possible without help and support of many people.

First of all, I would like to express my deepest gratitude to my thesis advisor, Dr. Robert P. Judd for his expertise, guidance, and continuous support during this study.

Second, I would like to thank my thesis committee, Dr. Gursel Suer, Dr. Dusan Sormaz and Dr. Faizul Huq, for their encouragement, insightful comments, and crucial questions.

Last but not the least; I would like to thank my parents and my dear siblings for their continuing love and support.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS AND DEFINITIONS

A	System Matrix
$\mathbf{A}^*$	Kleene star of **A**
DAG	Directed Acyclic Graph
CMS	Choice-free Manufacturing System
$G(N, E)$	Graph G with N and E as set of nodes and arcs respectively
$e(n, n') \in E$	The edge connecting nodes n to n'
$\text{pred}(n)$	$\{n' \in N \mid e(n', n) \in E\}$
$\text{succ}(n)$	$\{n' \in N \mid e(n, n') \in E\}$
$\oplus$	Maximization
$\otimes$	Addition
ε	Unit element of maximization $(= -\infty)$
e	Unit element of addition $(= 0)$.
$\mathbb{R}_{max}$	$\mathbb{R} \cup \{\varepsilon\}$
s	Start time vector
d	Due date vector
λ (**A**)	Eigenvalue of **A**
c	Completion time vector
l	Lateness vector
t	Tardiness vector
p	Cycle time or period of the system

ms	Makespan
$\mathbf{s}(n)$	Start time vector of node n
Q	Set of source nodes which is defined by $\{n \in N \mid \text{pred}(n) = \emptyset\}$
V	Counter
$t(n, n')$	The processing time of edge $e(n, n')$
$\mathbf{p}$	An array indexed by the nodes
$p(n)$	An index that is initialized with $\lvert\text{pred}(n)\rvert$
$\mathbf{u}_i$	Unit vector that is defined by $\begin{cases}[\mathbf{u}_i]_j = e, & i = j \\ [\mathbf{u}_i]_j = \varepsilon, & i \neq j\end{cases}$
$\mathbf{a}_i$	The i^{th} column of $\mathbf{A}$
$first(j)$	A function that returns the first node of the job j
$last(j)$	A function that returns the last node of job the j
J	Number of jobs

1. INTRODUCTION

It is important to have efficient algorithms to evaluate performance measurements in manufacturing systems. In order to analyze manufacturing systems from different aspects, considerable research has been done in recent years. Among these, sequencing and scheduling of jobs, machines, human resources and also customers all have been hot topics. In the scheduling domain, depending on the nature of the problem, there are different performance measures such as makespan, tardiness, etc. In order to tackle these problems, different mathematical modeling techniques have been used. This research, however, uses graph theory and max-plus algebra as mathematical tools that aid in the analysis of these systems.

The max-plus algebra representation of a system is given by the system matrix, **A**. Matrix **A** contains concise and useful information about the system. These types of information play a critical role in scheduling problems. It can be shown that **A** can be used to calculate different performance measurements such as makespan, critical path, cycle time, tardiness, lateness, etc. Although this research does not address the scheduling problem directly, it will make scheduling algorithms more efficient.

Kleene star of **A**, shown by $\mathbf{A}^*$, is another performance measure tool that contains longest path values. $\mathbf{A}^*$ is also used in solving linear max-plus algebra equations. Therefore, having an efficient algorithm to compute Kleene star of a matrix becomes important.

Proposed algorithms for calculating system matrix and Kleene star becomes more valuable if they can be applied for interval systems. In interval systems, input data such

as processing times are given by interval values. Therefore, at the end of this thesis, the algorithms are expanded to systems defined by intervals.

1.1. Research Scope

This thesis considers graphical representations of choice-free manufacturing systems to develop efficient algorithms for calculating system matrix and Kleene star, and analyzes the proposed algorithms in interval system. Choice-free manufacturing systems have predetermined schedule of jobs on machines. In other words, sequence of jobs on each machine and also sequence of machines on each job remains the same during each the production.

In addition, it will be assumed that the processing times are represented by either fixed times or by intervals. Since graph of the manufacturing systems under these assumptions has a special structure, called Directed Acyclic Graphs (DAG) [2], this research focuses on this type of graphs.

1.2.Contributions

- Developed an efficient algorithm to calculate start times of all operations in a manufacturing systems using dynamic programming and topology sorting simultaneously
- Developed an efficient and structured algorithm to calculate system matrix for any system with DAG structure
- Developed a method to calculate Kleene star operator for any system with DAG structure by using the system matrix algorithm that runs slightly faster than the fastest algorithm in the literature

- Performing interval analysis and expanding proposed algorithms to interval system

2. LITERATURE REVIEW

In order to develop efficient algorithms for calculating system matrix and Kleene star in manufacturing systems, this thesis uses max-plus algebraic techniques to model manufacturing systems. Then, it applies graph theoretical algorithms to come up with efficient algorithms.

Next sections provide a review of recent and relevant publications in the area of manufacturing systems and modeling them, dioids, max-plus as an example of dioids, system matrix, Kleene star, longest path problem and its relationship with system matrix and Kleene star, and finally, interval systems.

2.1. Manufacturing Systems

Modeling of manufacturing systems in order to analyze and optimize different performance measures is an important research topic. Imaev [3] developed graphical model to represent manufacturing systems. In his model, predetermined schedule of jobs and machines form a graph that jobs have horizontal flow on machines and machines have vertical flow on jobs. Moreover, each operation is exhibited by a node and sequence of resources described by directed arcs. When a buffer with infinite capacity is assumed for each operation, each of the nodes will have at most two input and two output arcs in the graphical model.

A manufacturing system with predetermined schedule of jobs and machines is called a Choice-free Manufacturing System (CMS) [4]. A graph of a CMS does not have any cycle, because existence of cycle means deadlock in the system [2]. Due to its special structure, CMS belongs to a certain type of graphs, called Directed Acyclic Graphs

(DAG). This special structure of CMS enables this study to use well-known graph theoretical algorithms for performance evaluation purposes.

This thesis uses DAGs to model manufacturing systems. Then, it applies longest path algorithm to develop efficient algorithms for calculating system matrix and Kleene star of the system.

2.1.1. Graphical Modeling of Manufacturing Systems

Given a Choice-free Manufacturing System (CMS), Imaev [5] showed that the system can be represented by stacking operation blocks together according the job specification and schedule. The resulting block diagram can be simplified into a DAG. To create the graph directly, just create a node for each operation. Then connect the nodes horizontally in the order that the corresponding operations are specified in the jobs. Likewise, the nodes are connected vertically as specified by the schedule. The edges (arcs) of the graph are assigned weights equal to the processing time of the operation. Notice, by this construction, every node has at most two incoming and outgoing arcs. Consider a manufacturing system described by the jobs in Table 1 and the sequence is given in Table 2. The graph representing this system is illustrated in Figure 1. For convenience, each operation is labeled with an integer.

Table 1: Processing order of jobs / Respective processing times

J1	M2/3	M1/4	M3/6
J2	M1/3	M2/4	M3/9
J3	M3/2	M2/1	M1/5

Table 2: Sequence for all the machines

M1	**M2**	**M3**
J2	J1	J3
J3	J2	J2
J1	J3	J1

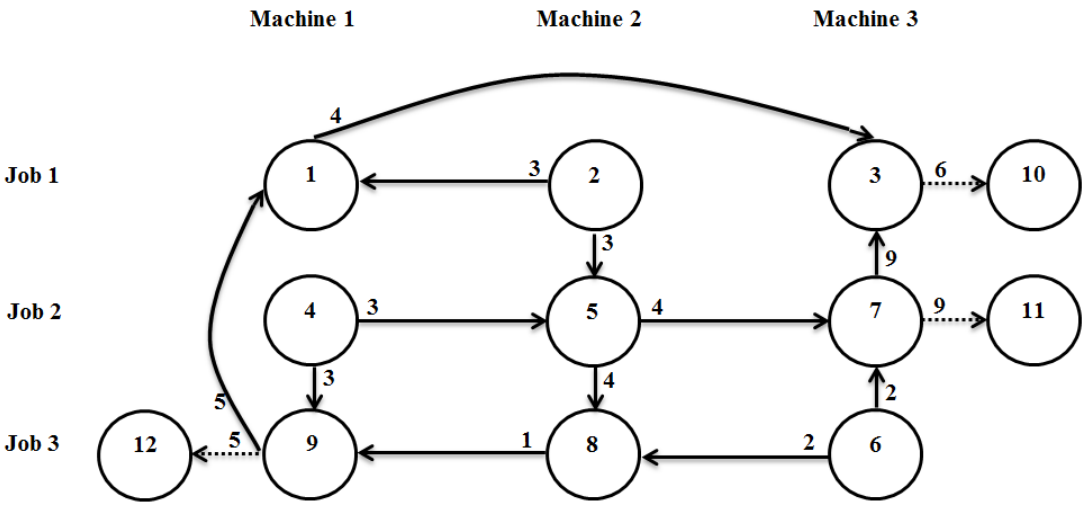

Figure 1: Graph of the given manufacturing system

In this representation, terminating node of each job is connected to a dummy node (shown by dashed arcs).

Let G be a DAG that corresponds to a manufacturing system. Define N to be the set of nodes and E the set of edges in G. Further, define $e(n_i, n_j) \in E$ to be the edge connecting nodes n_i to n_j. Define the predecessor and successor functions as follows

$$\text{pred}(n) = \{n' \in N | e(n', n) \in E\},$$

$$\text{succ}(n) = \{n' \in N | e(n, n') \in E\}.$$

A node n is called a *source* node if $\text{pred}(n) = \emptyset$; it is call a *destination* node if $\text{succ}(n) = \emptyset$; all other nodes are called *interior* nodes. Finally, define the function $t: E \to \mathbb{R}$ which maps the arcs to process times. These are the represented as weights of the arcs.

2.2.Dioids

Since all algorithms in this thesis are derived from longest path algorithms, which are developed over dioids, knowing the concept of dioid and its definition becomes important to understand the framework of proposed algorithms and also the way they are expanded to interval systems.

According to definition, set D equipped with addition ($\oplus$) and multiplication ($\otimes$) operations is called dioid if it has following properties [6]:

1- Associativity of addition

$$\forall\, a, b, c \in D, \qquad (a \oplus b) \oplus c = a \oplus (b \oplus c).$$

2- Commutativity of addition

$$\forall\, a, b \in D, \quad a \oplus b = b \oplus a.$$

3- Associativity of multiplication

$$\forall\, a, b, c \in D, \qquad (a \otimes b) \otimes c = a \otimes (b \otimes c).$$

4- Distributivity of multiplication with respect to addition

$$\forall\, a,b,c\ \in D\,, \qquad (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),$$

$$c \otimes (a \oplus b) = c \otimes a \oplus c \otimes b.$$

5- Existence of a null element

$$\exists \varepsilon \in D: \forall\, a\ \in D\,, \qquad a \oplus \varepsilon = a\,.$$

6- Absorbing null element

$$\forall\, a\ \in D\,, \qquad a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon\,.$$

7- Existence of an identity element

$$\exists e \in D: \forall\, a\ \in D\,, \qquad a \otimes e = e \otimes a = a\,.$$

8- Idempotency of addition

$$\forall\, a\ \in D\,, \qquad a \oplus a = a\,.$$

2.2.1. Max-plus Algebra

Max-plus algebra, as an example of a dioid [6], is used in this thesis to develop efficient algorithms for calculating system matrix and Kleene star operator. This section provides background research in max-plus, then it introduces the basics of this algebra that are used in this thesis.

2.2.1.1.Background of Max-plus Algebra

Regular algebra is equipped with two operations, addition and multiplication. In max-plus algebra, as its name indicates, there are maximization and addition operators. Max-plus algebra is a tool to model discrete event systems and to analyze marking times in petri nets[6]. A petri net is a graphical representation of a system that can be used to evaluate the system dynamically. Therefore, most researchers not only use block

diagrams, state equations, and queueing theory, but also petri nets and max-plus algebraic tools to model and analyze manufacturing systems.

In the scheduling of manufacturing systems with blockings, Mejia [7] used petri net approach to model the system and evaluate the scheduling issues. In addition, Labadi [8] used stochastic petri net models to modeling and performance analysis of logistic systems. Moreover, modeling a transportation system with petri nets and analyzing scheduling time tables with max-plus algebra were introduced in [9]. Basics of max-plus algebra, algorithms to calculate eigenvectors and eigenvalues and their interpretations in max-plus algebra have been studied in [9], [10].

For scheduling and performance evaluations purposes, max-plus operators can be used to simplify the system complexities. For example, an operation can be started when both required machine and part are available, therefore, starting time of the operation is simply the maximum of the time that both machine and part are available. To complete the operation, one can also add its associated processing time by an addition operator. Because of this simplicity and convenience, significant research has applied max-plus algebra in the scheduling area which some of them are mentioned in the following paragraphs.

Minimizing product of matrices by max-plus algebra and its application in scheduling problems such as single machine, two- and three-machine flowshop scheduling problems have been studied in [11]. Considering makespan as objective function and analyzing it with max-plus algebra can be found in [12], [13], [14], [15]. In these papers, jobshop scheduling problem with and without recirculation conditions were

taken into account. In addition, researchers focused on flowshop scheduling problem and evaluated different performance analysis. For example, Imaev [16] proposed algorithms to explain spectral properties for the max-plus dynamics matrix for flowshops. Moreover, Nambiar [17] developed mathematical formulation for cyclic permutation flowshops using max-plus algebra. In this type of problem, a group of certain parts is repeatedly produced within a fixed cycle. In all cycles, the sequence of machines and jobs remains the same. He proposed an efficient formulation to compute the period of this type of scheduling systems [17]. On the other hand, max-plus algebra and its application in cyclic jobshop scheduling problem has been studied in [18].

Max-plus linear equations can be used to control and analyze flow line by generating state-space equations. Seleim modeled manufacturing flow lines with considering two scenarios, with finite buffer capacity and with infinite buffer capacity [19], [20].

Kajiwara studied max-plus algebraic techniques to scheduling problems of block assembly line [21]. Other researchers applied max-plus algebraic algorithms to schedule ship building lines [22], [23].

Model predictive scheduling of semi-cyclic discrete event systems by max-plus linear models were also studied by Van den [24]. He considered system matrix and studied control of routing, ordering and synchronization issues by a set of variables.

High-variety and low-volume manufacturing systems and their scheduling problems under the condition of preventive maintenance has been studied in [25].

Alirezaei studied max-plus algebra and its application in optimal scheduling of multiple sheets in a printer [26].

Max-plus algebra also has been used frequently to analyze different queueing systems. Stochastic processes and Markov diffusion processes in max-plus algebra have been studied in [27]. Max-plus modeling and its application to analyze different queueing systems have been studied in [28], [29], [30]. Monitoring and performance evaluation of specific classes of queueing systems have been studied in [31], [32].

2.2.1.2.Definitions and Notations

In max-plus algebra, as its name indicates, there are maximum and addition operators which are represented by $\oplus$ and $\otimes$ respectively. The unit element for maximization is defined by $\varepsilon(=-\infty)$ and the unit element of addition is $e\ (=\ 0)$. Define $\mathbb{R}_{max}$= $\mathbb{R} \cup \{\varepsilon\}$ and let $x, y \in \mathbb{R}_{max}$, the two aforementioned operators are defined as follows

$$x \oplus y\ = \max(x, y),$$

$$x \otimes\ y = x + y\ .$$

Let **A** and **B** be two $m \times n$ matrices in $\mathbb{R}_{max}^{mxn}$, then the addition of two matrices is defined as follows

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

which right side takes the maximum of ij elements in **A** and **B** matrices and saves as the ij element of the **A** maximum **B** matrices.

Multiplication operator between matrices $\mathbf{A} \in \mathbb{R}_{max}^{mxn}$ and $\mathbf{B} \in \mathbb{R}_{max}^{nxo}$ is defined as

$$[\mathbf{A} \otimes \mathbf{B}]_{i,j} = \oplus_{k=1}^{n} ([\mathbf{A}]_{i,k} \otimes [\mathbf{B}]_{k,j})\ .$$

Finally, for vectors $\mathbf{a}, \mathbf{b}$ in $\mathbb{R}_{max}^{n}$ and scalar c we define the following vector operations

$$[\mathbf{a} \oplus \mathbf{b}]_i = [\mathbf{a}]_i \oplus [\mathbf{b}]_i \ .$$

$$[\mathbf{a} \otimes \mathrm{c}]_i = [\mathbf{a}]_i \otimes c \ .$$

2.3.System Matrix

System matrix contains brief and useful information about the system which can be used for performance evaluation purposes [9]. The max-plus algebra representation of system is represented by the system matrix, **A**. The ij element of **A** represents the completion times of job i , given that job j is started at time zero and all other jobs started early enough that do not affect job i (theoretically, this time is $-\infty$) [14].

A few researchers studied algorithms to generate the state-space equations of the system using max-plus algebra [20, 33-35]. These equations contain the same information that system matrix does; however, using set of equations for evaluating and controlling manufacturing systems is burdensome.

From the computational complexity perspective of system matrices, as Singh [14] showed, a manufacturing system can be represented by transition matrices **B**, **C** and **D**, input vector **u** and output vector **y** as shown in Figure 2.

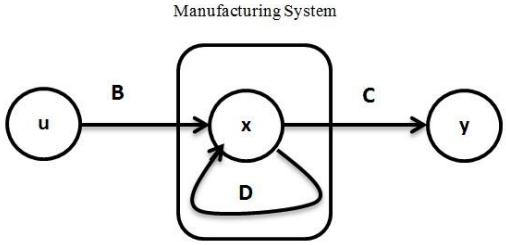

Figure 2: General graph representation of a manufacturing system

Defining max-plus algebra equations [9] for the system represented by Figure 2, results

$$\mathbf{x} = \mathbf{D} \otimes \mathbf{x} \oplus \mathbf{B} \otimes \mathbf{u}\,,$$

$$\mathbf{y} = \mathbf{C} \otimes \mathbf{x} = \mathbf{C} \otimes (\mathbf{D} \otimes \mathbf{x} \oplus \mathbf{B} \otimes \mathbf{u}) = \mathbf{C} \otimes \mathbf{D}^{*} \otimes \mathbf{B} \otimes \mathbf{u}\,.$$

According to definition, system matrix is a matrix that relates input vector to the output vector; therefore,

$$\mathbf{A} = \mathbf{C} \otimes \mathbf{D}^{*} \otimes \mathbf{B}\,.$$

The star operator in $\mathbf{D}^{*}$ is called Kleene star.

This thesis uses the concept of longest path in DAG structures to develop efficient algorithms for calculating system matrix and Kleene star.

From the application perspective, if $\mathbf{s}$ (start time vector), $\mathbf{d}$ (due date vector), $\mathbf{A}$ (system matrix), and λ (eigenvalue of $\mathbf{A}$); then, completion time vector ($\mathbf{c}$), lateness vector ($\mathbf{l}$), tardiness vector ($\mathbf{t}$), cycle time or period of the system (p), and makespan (ms) can be calculated as followings

$$\mathbf{c} = \mathbf{A} \otimes \mathbf{s}\,,$$

$$\mathbf{l} = \mathbf{d} - \mathbf{A} \otimes \mathbf{s}\,,$$

$$\mathbf{t} = (\mathbf{d} - \mathbf{A} \otimes \mathbf{s}) \oplus \mathbf{e}\,,$$

$$p = \lambda\,(\mathbf{A}),$$

$$ms = \oplus_{i=1}^{i=J} \oplus_{j=1}^{j=J} a_{ij}\,.$$

For example, consider a manufacturing system represented in Figure 1. System matrix of this system is $\mathbf{A}=\begin{bmatrix}23 & 23 & 18\\ 16 & 16 & 11\\ 13 & 13 & 8\end{bmatrix}$. From the application perspective, assume that start time vector is $\mathbf{s} = \begin{bmatrix}\varepsilon\\ \varepsilon\\ e\end{bmatrix}$, then completion time of jobs are $\mathbf{c} = \mathbf{A} \otimes \mathbf{s} = \begin{bmatrix}23 & 23 & 18\\ 16 & 16 & 11\\ 13 & 13 & 8\end{bmatrix} \otimes \begin{bmatrix}\varepsilon\\ \varepsilon\\ e\end{bmatrix} = \begin{bmatrix}18\\ 11\\ 8\end{bmatrix}$ which is shown in the following gantt chart.

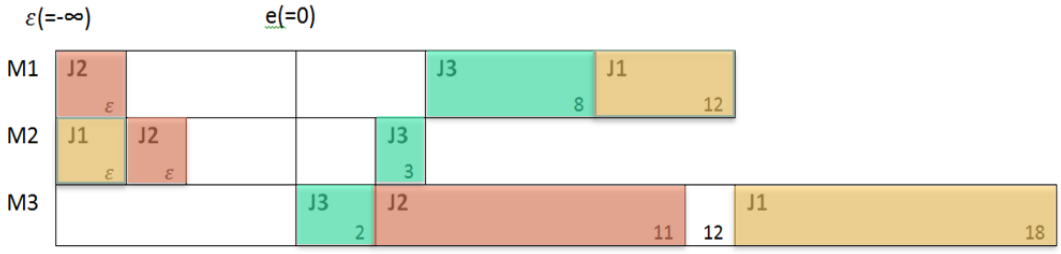

Figure 3: Gantt chart of the system for $\mathbf{s} = \mathbf{u_3}$

As another illustration of system matrix application, assume that all jobs start at time zero, then $\mathbf{s} = \begin{bmatrix}e\\ e\\ e\end{bmatrix}$ and $\mathbf{c} = \mathbf{A} \otimes \mathbf{s} = \begin{bmatrix}23 & 23 & 18\\ 16 & 16 & 11\\ 13 & 13 & 8\end{bmatrix} \otimes \begin{bmatrix}e\\ e\\ e\end{bmatrix} = \begin{bmatrix}23\\ 16\\ 13\end{bmatrix}$. Figure 4 represents its gantt chart.

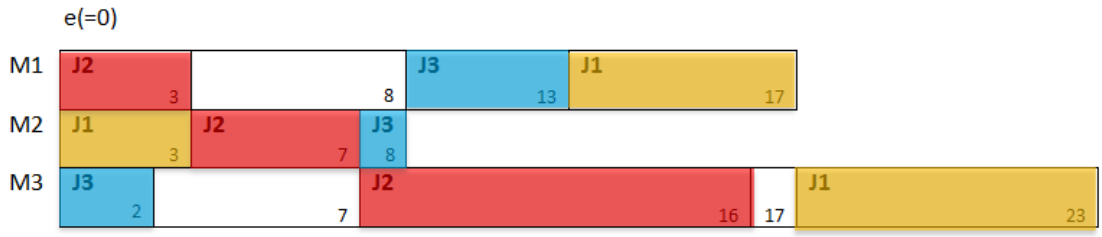

Figure 4: Gantt chart of the system for **s** = **e**

Finally, the makespan of the system is equal to the maximum element of the system matrix which gives $ms = 23$.

2.4.Kleene Star

Kleene star of a matrix, shown by $\mathbf{A}^*$, is an important operator in max-plus algebra. Its primary purpose is in solving linear max-plus algebraic equations [9]. For matrix $\mathbf{A} \in \mathrm{R}_{max}^{nxn}$, it is defined as

$$\mathbf{A}^* = \bigoplus_{\mathbf{i=0}}^{\infty} \mathbf{A}^{\otimes \mathbf{i}} = \mathbf{E} \oplus \mathbf{A}^1 \oplus \mathbf{A}^2 \oplus \mathbf{A}^3 \ldots .$$

It is shown [9] that if $\mathbf{A}^*$ exists, the maximization needs only go through $i = n - 1$ where n is the number of columns in $\mathbf{A}$. Calculating Kleene star with this definition requires

$O(2n^4 - 2n^3 + 2n^2)$ operations, because each element of $\mathbf{A}^2$ requires $2n - 1$ computations ($n - 1$ times of $\oplus$, and n times of $\otimes$), therefore, $2n^3 - n^2$ computations are required for computing $\mathbf{A}^2$. Calculating all other powers require the same computation. Since there are $n - 2$ different powers (starting from 2 to $n - 1$), $2n^4 - 3n^3 + 2n^2$ computations are needed for generating all powers. After that, maximization operator compares all elements of these matrices which needs n^3 computations

(n^2operation required for comparing two matrices, and there are n different matrices). Therefore, this algorithm has total complexity of $O(2n^4 - 2n^3 + 2n^2)$.

A bit more efficient algorithm is called divide and conquer which divides the matrix into four sub-matrices.

$$\mathbf{A}^* = \begin{bmatrix} \mathbf{K_1} & \mathbf{K_2} \\ \mathbf{K_3} & \mathbf{K_4} \end{bmatrix}^* = \begin{bmatrix} \mathbf{K_1^*} \oplus \mathbf{K_1^* K_2 (K_3 K_1^* K_2} \oplus \mathbf{K_4)^* K_3 K_1^*} & \mathbf{K_1^* K_2 (K_3 K_1^* K_2} \oplus \mathbf{K_4)^*} \\ \mathbf{(K_3 K_1^* K_2} \oplus \mathbf{K_4)^* K_3 K_1^*} & \mathbf{(K_3 K_1^* K_2} \oplus \mathbf{K_4)^*} \end{bmatrix}$$

Using this method recursively results the Kleene star of the system in a complexity of $\frac{8}{3}O(n^3)$ [14], [36].

Goto [37, 38] proposed an algorithm to calculate Kleene star for graph representation of a system. This algorithm assumes that topological sort of the graph is given. Based on the sorted graph, it computes the Kleene star. In terms of complexity, this algorithm has two parts that need to be summed. First part is topology sorting of the graph, and the second part is the algorithm. Since Kahn's algorithm is the most efficient way to sort the nodes topologically, this research considers Kahn algorithm's complexity for the first part which is $O(7|N| + 3|E|)$.

The second part of the algorithm is similar to the steps 2-6 of algorithm 2. Therefore, total complexity is $((2|N| + 4)|E| + |N|^2 + 9|N|)$.

He also studied computing Kleene star in a fast way by using parallelization concept in cell broadband engine [39]. Moreover, he designed numerical examples to illustrate the efficiency of this methodology. In addition, he introduced a fast way to calculate Kleene star in max-plus algebra using Compute Unified Device Architecture (CUDA) graphic processing units [40].

2.5. Longest Path Problem

For a given graph of a manufacturing system, longest path is the same as makespan of the system [13]. There are multiple ways to compute the longest path within a graph. One of the ways to find longest path in graphs is to multiply the distances by -1 and find the shortest path by using shortest path algorithms. Following paragraphs review shortest path algorithms and their applicability to this idea.

One of the most common algorithms for finding shortest path within a graph is introduced by Dijkstra [41]. This algorithms finds the shortest path from a given source node to each of the other nodes with time complexity of $O(|N|^2)$ which N is the set of graph nodes [42]. Since Dijkstra algorithm is a greedy algorithm [42], it is not possible to use negative distances for calculating longest path.

Another shortest path algorithm is developed separately by three researchers, Bellman, Ford and Moore [43-45]. Therefore, it is known as Bellman-Ford-Moore algorithm. This algorithm runs in $O\ (3|N||E| + 3|E| + 3|N|)$ time which E is the set of edges. This algorithm handles edges with negative weights and uses dynamic programming, therefore, it is possible to modify this algorithm to find the longest path [43-45].

In addition, Floyd-Warshal algorithm finds shortest path between all pairs of nodes in $O(2|N|^3 + |N|^2 + 2|E| + 2|N|)$ time [46]. This algorithm also handles edges with negative weights. It is an example of dynamic programming; therefore, it can be applied for finding longest path [47].

The longest path can also be obtained by sorting nodes topologically. Topology sort or topology order of nodes, is a process that makes a sequence of nodes in a line in which predecessors of each node appears before that node itself [1]. Researchers studied topology sort problem in order to develop efficient algorithms to generate them. Er [48] studied parallel computation method to generate topology orders.

Since topology sort is not unique and there might be multiple topology sorts for a given graph, Kalvin [49] proposed an algorithm that generates all topology orders of a given graph.

Considering DAG structure of manufacturing systems, Pearce [50] introduced a simple algorithm that dynamically maintains the topology order of DAG's. The simplicity of his proposed algorithm improves the running time of the algorithm.

Inoue [51] introduced an algorithm to generate all topology sorts of a DAG based on data structure and permutations of nodes. They proposed the first algorithm for implicit generation of all topology sorts with dynamic manipulation in the most efficient way in terms of time and space [51].

Italiano [52] reviewed different topology sort algorithms and showed that Kahn's algorithm [1] is the most efficient algorithm to find a single topology sort in a DAG structure [53]. Therefore, this thesis uses Kahn's algorithm as a base to develop most efficient algorithms to calculate system matrix and Kleene star.

The most efficient way to calculate the longest path is to sort the nodes topologically and use dynamic programming [37]. Topological ordering of a directed

graph is a linear ordering of its nodes such that for every directed arc $e(n, n')$ from node n to node n', n comes before n' in the ordering.

Sorting the nodes take place based on their priorities. Therefore, it is impossible to find a topological sort for a graph if there is a cycle, because any node in the cycle is both predecessor and successor of other nodes in the same cycle. As a summary, topology sort exists for only DAGs. Obviously, the topological sort is not unique if it exists.

The most efficient way for topological sorting is introduced by Kahn [1], [53]. Since manufacturing systems are DAG, this thesis uses this algorithm to sort the nodes topologically. After doing that, longest path algorithm is calculated using dynamic programming.

2.6.Interval System

A system with defined operations on intervals is called an interval system. Interval values are represented by pair $[a, b]$ that indicates $a \leq x \leq b$ [54]. Interval analysis can be used for analyzing a system and finding optimal solutions. For example, Schichl [55] applied interval analysis on DAGs for finding global optimum solutions. Different methods and applications of interval systems can be found in [56].

In this thesis, proposed algorithms are expanded to interval systems where the input values of algorithms are considered intervals.

3. SYSTEM MATRIX

System matrix (**A**) is square matrix that connects the input vector to the output vector. System matrix contains useful data about the time delays between different operations; therefore, it can be used to calculate and analyze different performance measurements in a manufacturing system such as makespan, tardiness, lateness, etc.

In order to calculate system matrix efficiently, this thesis modifies Kahn's algorithm to include the dynamic programming required to compute start and completion times of jobs, then it modifies the algorithm to calculate system matrix.

3.1. Modified Algorithm to Calculate Start Times

Algorithm 1 uses Kahn's [1] topological sort algorithm and dynamic programming at the same time to calculate start times.

Define $s(n)$ as the start time of node n. The set Q is initialized with all the source nodes. Step 9 only adds a node to the set Q if all of its predecessors have been visited. Step 6 will update $s(n')$, where n' is a successor of node n. Notice that when node n' is added to Q, $s(n')$ will be the maximum of all its predecessors. Hence, it would then be the start time of the operation.

Steps 11-12 detect the cycles if there is any. The counter V (number of visited nodes) is initialized with the number of nodes in the set Q. Every time that a node gets added to Q, this counter is incremented. After the algorithm terminates, if some of the nodes have not been added to Q, then a cycle exists which means that DAG represents a system with an infeasible solution.

Algorithm 1: Modification of Kahn algorithm for topological sort to calculate the start time of each node

input DAG
$t(n, n')$ is the processing time of edge $e(n, n')$
output $s(n)$ is the start time of each node

1. **let** Q be a set of nodes, and **p** an array indexed by the nodes
2. **initialize** $p(n) = |\text{pred}(n)|$, $s(n)$ = ε, $\forall n$; and $Q = \{n \in N | \text{pred}(n) = \emptyset\}$, V=$|Q|$
3. **while** Q **is not** empty
4. **remove** node *n* **from** Q
5. **for all** $n' \in \text{succ}(n)$
6. $s(n') = s(n') \oplus s(n) \otimes t(n, n')$
7. **decrement** $p(n')$
8. **if** ($p(n')$=0) **then**
9. **add** n' to Q
10. **increment** V
11. **if** (V< $|N|$) **then**
12. **return** ERROR: Infeasible Solution

Each node in the graph enters to Q at most once (if there is no cycle in the graph, each node enters once; otherwise, the nodes in the cycle do not enter). When a node enters to Q, it will finally exit from Q by running step 4 of the algorithm. Therefore, this algorithm terminates.

Moreover, when a node is removed from Q, $s(n)$ is set to the maximum of the completion time of all of its predecessor nodes (step 4-6). The completion time of an operation is the sum of its start time plus its operation time. Thus $s(n)$ is the start time of node n. Therefore, when the algorithm completes, $s(n)$ will contain the start time of all the operations given that source nodes start their operations at time 0.

The complexity of the algorithm 1 is fairly easy to compute. Going back to algorithm 1, step 2 initializes $2|N|$ elements; each node is added to Q only once, so steps 3-4 and 9-10 are each executed $|N|$ times. Steps 5, through 8 are execute for every

outgoing arc for every node, or simply every arc in G; hence, there are $5|E|$ operations in these steps. For any feasible schedule, steps 11-12 require only one operation.

Therefore, there is a total of $6|N| + 5|E|$ operations, since $|E| \cong 2|N|$ for large manufacturing systems, the complexity of the algorithm is $O(16|N|)$ for DAG that represent such systems. Since each operation is represented by a node, the algorithm in linear with respect to the number of operations in the system.

3.2. System Matrix Algorithm

Define the unit vector

$$\mathbf{u}_i = \begin{cases} [\mathbf{u}_i]_j = e, \ i = j \\ [\mathbf{u}_i]_j = \varepsilon, \ i \neq j \end{cases}.$$

Then it is easy to verify that

$$\mathbf{a}_i = \mathbf{A} \otimes \mathbf{u}_i \ ,$$

where $\mathbf{a}_i$ is the i^{th} column of $\mathbf{A}$. So, instead of initializing all the jobs to start at time 0 (i.e e) as was done implicitly in Algorithm 1, only job i starts at time 0 and the others start at ε. Then the start time of the last nodes (the nodes that are added to the original graph) for each job form the elements of the column $\mathbf{a}_i$. From an algorithmic perspective, this can be accomplished easily by initializing

$$s(n) = \begin{cases} \mathrm{e} & n = \mathrm{first}(j) \\ \varepsilon & \text{otherwise} \end{cases},$$

where $\mathrm{first}(j)$ is a function that returns the starting node for job j and then executing Algorithm 2. Then if $\mathbf{a}_i$ is the i^{th} column of the system matrix associated with job j, then

$$[\mathbf{a}_i]_j = c\big(\mathrm{last}(j)\big)^T \ ,$$

where $\mathrm{last}(j)$ is a function that returns the terminating node for job j.

By executing the algorithm multiple times with initial values corresponding to $\mathbf{u}_j$ $1 \leq j \leq J$ (J is the number of jobs) and collecting the start times of the last operation of all jobs then the entire matrix $\mathbf{A}$ is found. This results in a total complexity of $O\ (16\,J|N|)$ to find the system matrix where N is the set of nodes.

A slightly more efficient algorithm can be developed, by realizing that the topological sort does not need to be regenerated for each job. In the new algorithm each node will hold a vector $\mathbf{s}(n)$ of start times, where the j^{th}component is the start time of the node assuming the initial starting times of the jobs are initialized to $\mathbf{u}_j$. These vectors are initialized as follows

$$\mathbf{s}(n) = \begin{cases} \mathbf{u}_j & n = \text{first}(j) \\ \boldsymbol{\varepsilon} & \text{otherwise} \end{cases},$$

where $\boldsymbol{\varepsilon}$ is a vector of the additive identity elements.

As similar to algorithm 1, steps 11-12 detect the cycles if there is any by counting the number of entered nodes to Q. For any feasible schedule that does not contain any cycle in its graph, these steps do not return any error. The counter V (number of visited nodes) initializes with the number of nodes in the set Q. Every time that a node gets added to Q, this counter becomes incremented. After the algorithm terminates, if some of the nodes are not added to Q, it means an infeasible schedule.

Algorithm 2: Modification of Algorithm 1 to compute **A**

input DAG
$t(n,n')$is the processing time of edge $e(n,n')$
output system matrix

1. **let** Q be a set of nodes, and **p** an array indexed by the nodes
2. **initialize** $p(n) = |\text{pred}(n)|$, $\mathbf{s}(n) = \begin{cases} \mathbf{u}_j & n = \text{first}(j) \\ \boldsymbol{\varepsilon} & \text{otherwise} \end{cases}$, and $Q = \{n \in N | \text{pred}(n) = \emptyset\}$, V=$|Q|$
3. **while** Q **is not** empty
4. **remove** node *n* **from** Q
5. **for all** $n' \in \text{succ}(n)$
6. $\mathbf{s}(n') = \mathbf{s}(n') \oplus \mathbf{s}(n) \otimes t(n,n')$
7. **decrement** $p(n')$
8. **if** ($p(n')$=0) **then**
9. **add** n' to Q
10. **increment** V
11. **if** (V< $|N|$) **then**
12. **return** ERROR: Infeasible Solution
13. **return** $\mathbf{A} = \begin{bmatrix} \mathbf{s}(\text{last}(1))^{\mathrm{T}} \\ \mathbf{s}(\text{last}(2))^{\mathrm{T}} \\ \mathbf{s}(\text{last}(3))^{\mathrm{T}} \\ \vdots \\ \mathbf{s}(\text{last}(\mathrm{J}))^{\mathrm{T}} \end{bmatrix}$

Again, each node in the graph enters to Q at most once (if there is no cycle in the graph, each node enters once; otherwise, the nodes in the cycle do not enter). When a node enters to Q, it will finally exit from Q by running step 4 of the algorithm. Therefore, this algorithm terminates.

In addition, when a node is removed from Q, the operation times of its successor edges are added to $\mathbf{s}(n)$ and compared with $\mathbf{s}(n)$ of related successor (step 4-6). Thus when the algorithm completes, $\mathbf{s}(n)$ will contain the start time vectors. Since dummy nodes are directly added to the end of each job by an edge with weight zero, the start time

of the dummy nodes are the same as completion times of their previous nodes. Therefore, $\mathbf{s}(n)$ of last jobs determine the rows of the system matrix.

In terms of computational complexity, each of the steps 3-4-9-10 requires $|N|$ computations. Step 2 requires $|N| + J|N|$ initializations. Similar to algorithm 1, each of the steps 5-7-8 requires $|E|$ computations and step 6 needs $2J|E|$ computations. Step 11 requires only one computation and step 12 does not require any computation for feasible schedules. Finally, step 13 returns the system matrix. This step does not require any computation if $\mathbf{s}(n)$ value of terminating node in each job is stored in this format from the beginning. Therefore, the total complexity is$O(2J|E| + J|N| + 3|E| + 5|\mathrm{N}|)$. In large systems, $|E| \cong 2|N|$, hence, total complexity is $O(5J|N| + 11|\mathrm{N}|)$.

Therefore, $O(5J|N| + 11|\mathrm{N}|)$ compares favorably to the complexity $O(16J|N|)$, because in large systems, $5J|N| \gg 11|\mathrm{N}|$; hence, complexity $O(5J|N|)$ shows significant improvement, over three times faster.

3.3.Numerical Example

Consider the system modeled by the DAG in Figure 1. First the **s(*n*)** vectors are initialized as shown in Table 3.

Table 3: Initial $\boldsymbol{s}(\boldsymbol{n})$ vectors

Node	1	2	3	4	5	6	7	8	9
Initial Value for **s**(n)	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}e\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ e\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ e\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$	$\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}$

The set Q is initialized to $\{2, 4, 6\}$. Suppose arbitrarily node 2 is removed from Q. The successors of node 2 are 1 and 5. So step 7 results in

$$\mathbf{s}(1)=\begin{bmatrix}(\varepsilon \oplus 3)\\(\varepsilon \oplus \varepsilon)\\(\varepsilon \oplus \varepsilon)\end{bmatrix}=\begin{bmatrix}3\\\varepsilon\\\varepsilon\end{bmatrix} \text{ and } \mathbf{s}(5)=\begin{bmatrix}(\varepsilon \oplus 3)\\(\varepsilon \oplus \varepsilon)\\(\varepsilon \oplus \varepsilon)\end{bmatrix}=\begin{bmatrix}3\\\varepsilon\\\varepsilon\end{bmatrix}.$$

After removing node 4 from Q and executing steps 4-8, node 5 becomes eligible to be added to Q, since $p(5)$ is now 0. Hence, Q now contains $\{5, 6\}$. This process repeats until Q gets empty. Table 4 summarizes the results of running algorithm 2.

Table 4: Algorithm results for each n exited from Q

Q	node removed from Q	Updated values of **s**(n)
2,4,6	2	**s**(1) = $\begin{bmatrix}3\\ \varepsilon\\ \varepsilon\end{bmatrix}$, **s**(5)= $\begin{bmatrix}3\\ \varepsilon\\ \varepsilon\end{bmatrix}$
4,6	4	**s**(5)= $\begin{bmatrix}3\\ 3\\ \varepsilon\end{bmatrix}$, **s**(9)= $\begin{bmatrix}\varepsilon\\ 3\\ \varepsilon\end{bmatrix}$
5,6	5	**s**(7)= $\begin{bmatrix}7\\ 7\\ \varepsilon\end{bmatrix}$, **s**(8)= $\begin{bmatrix}7\\ 7\\ \varepsilon\end{bmatrix}$
6	6	**s**(7) = $\begin{bmatrix}7\\ 7\\ 2\end{bmatrix}$, **s**(8)= $\begin{bmatrix}7\\ 7\\ 2\end{bmatrix}$
7,8	7	**s**(3)= $\begin{bmatrix}16\\ 16\\ 11\end{bmatrix}$, **s**(11)= $\begin{bmatrix}16\\ 16\\ 11\end{bmatrix}$
8	8	**s**(9)= $\begin{bmatrix}8\\ 8\\ 3\end{bmatrix}$
9	9	**s**(1)= $\begin{bmatrix}13\\ 13\\ 8\end{bmatrix}$, **s**(12)= $\begin{bmatrix}13\\ 13\\ 8\end{bmatrix}$
1	1	**s**(3)= $\begin{bmatrix}17\\ 17\\ 12\end{bmatrix}$
3	3	**s**(10) = $\begin{bmatrix}23\\ 23\\ 18\end{bmatrix}$

The system matrix can be found as follows

$$\mathbf{A}=\begin{bmatrix}\boldsymbol{s}(last(1))^T\\ \boldsymbol{s}(last(2))^T\\ \boldsymbol{s}(last(3))^T\end{bmatrix}=\begin{bmatrix}23 & 23 & 18\\ 16 & 16 & 11\\ 13 & 13 & 8\end{bmatrix}$$

4. KLEENE STAR OPERATOR

First section of this chapter provides a new approach to compute Kleene star based on dynamic programming and the concept of system matrix algorithm that is proposed in this thesis. Second section provides a numerical example for the proposed algorithm in section one. Third section overviews different graphical and algebraic approaches and compares them in terms of computational complexity.

4.1. Kleene Star Algorithm

In general graph representation of manufacturing systems, inputs of the system (**x**) is connected to outputs (**y**) with system matrix (**A**). By connecting outputs (**y**) and also auxiliary nodes (**u**) to inputs **(x)** with identity matrix and generating max-plus equations, Kleene star can be computed.

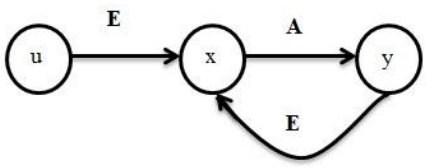

Figure 5: Graph representation of $\mathbf{A}^*$

$$\mathbf{y} = \mathbf{A} \otimes \mathbf{x},$$

$$\mathbf{x} = \mathbf{u} \oplus \mathbf{y} = \mathbf{u} \oplus \mathbf{A} \otimes \mathbf{x}.$$

The last equation is solved by $\mathbf{x} = \mathbf{A}^* \otimes \mathbf{u}$ [9]. In this equation, $\mathbf{A}^*$ connects the inputs to the outputs. Therefore, system matrix of Figure 5 is the same as Kleene star of

A. Figure 6 shows the expanded form of a system with arc weights. Elements of matrix **A** are the weights for continuous arcs, yet dashed arcs are all weighted with e.

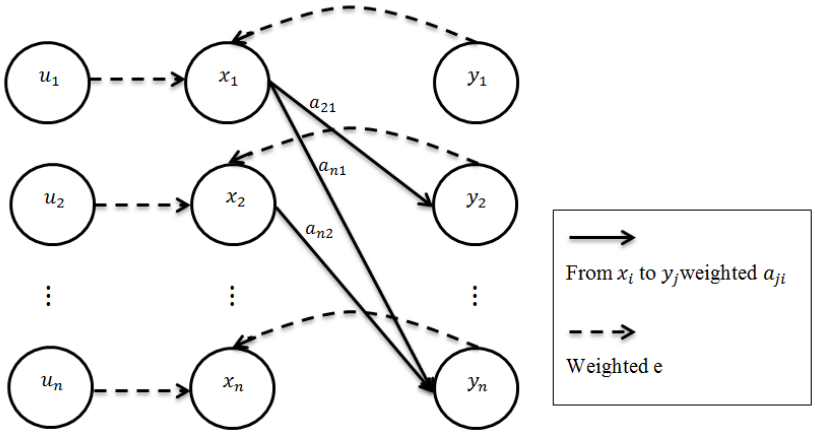

Figure 6: Expanded graph representation of $\mathbf{A}^*$

Since u_i and y_i nodes directly connected to input nodes with weight e, these nodes can be eliminated from the graph. Figure 7 represents the final form of the system after removing these nodes.

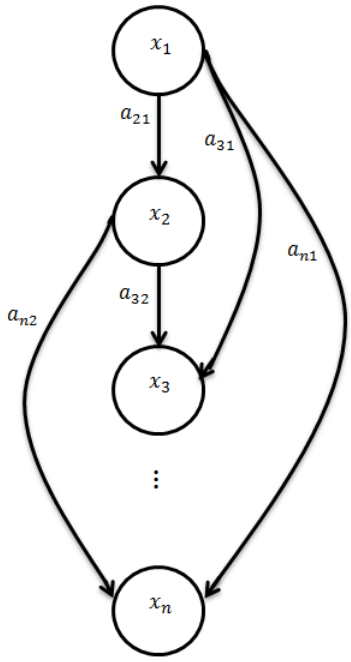

Figure 7: Final graph representation of a $\mathbf{A}^*$

In terms of computational complexity, this algorithm has $O(2J|E| + J|N| + 3|E| + 5|\mathrm{N}|)$ complexity which is the same as algorithm 2.

According to Figure 7, $J = |N|$. Therefore, complexity of the proposed algorithm is $((2|N| + 3)|E| + |N|^2 + 5|N|)$.

Table 5 summarizes the introduced algorithms for calculating Kleene star of **A** along with their computational complexities. In first two algorithms that run with matrix data, complexity is described based on the number of rows or columns; however, in last two algorithms that run with graph, complexity is mentioned based on the number of nodes in graph. For a system that is described with both matrix and graph forms, $|N| = n$

, therefore, Table 5 uses n for convenience. In addition, one may consider $|E| \leq \frac{|N|(|N|-1)}{2}$ for the last two algorithms and compare their complexities with first two ones.

Table 5: Comparison of computational complexity of different algorithms for A*

Algorithm	**Data Type**	**Complexity**
Basic Definition	Matrix	$O(2n^4 - 2n^3 + 2n^2)$
Divide and conquer	Matrix	$\frac{8}{3}O(n^3)$
Goto	Graph	$O((2n+4)\|E\| + n^2 + 9n)$
Proposed in this thesis	Graph	$O((2n+3)\|E\| + n^2 + 5n)$

4.2.Numerical Example

Figure 8 represents a system with three jobs where job indices are shown within nodes. In addition, $\mathbf{A} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ 5 & \varepsilon & \varepsilon \\ 3 & 4 & \varepsilon \end{bmatrix}$ represents the same system in a matrix form.

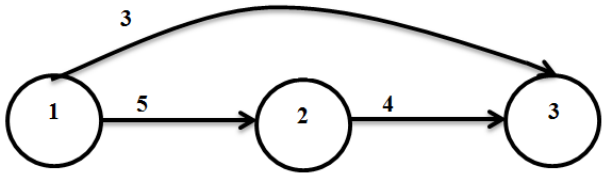

Figure 8: Graph of a system

In order to use Algorithm 2 to compute Kleene star, the system needs to be shown in the form of Figure 7. Figure 9 represents the system in this form where each row represents a job.

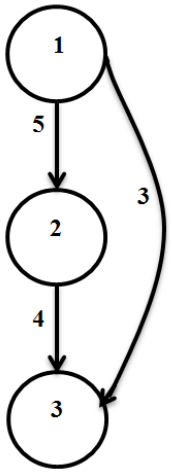

Figure 9: Converted graph of A

Considering the equation $\mathbf{x} = \mathbf{A}^* \otimes \mathbf{u}$ which is generated earlier, $\mathbf{u}$ nodes are inputs, $\mathbf{x}$ nodes are outputs, and $\mathbf{A}^*$ is the system matrix. Since $\mathbf{u}$ nodes are unified with $\mathbf{x}$ nodes (shown in Figure 6 and Figure 7), $\mathbf{x}$ nodes are both first nodes and last nodes in each row. Following this facts, $\mathbf{s}(n)$ vectors are initialized in Table 6.

Table 6:Initial $\boldsymbol{s(n)}$values

Node	1	2	3
Initial Value for $\mathbf{s}(n)$	$\begin{bmatrix} e \\ \varepsilon \\ \varepsilon \end{bmatrix}$	$\begin{bmatrix} \varepsilon \\ e \\ \varepsilon \end{bmatrix}$	$\begin{bmatrix} \varepsilon \\ \varepsilon \\ e \end{bmatrix}$

Table 7 summarizes the results of running algorithm 2 with DAG, shown in Figure 9.

Table 7: Algorithm results for each *n* exited from *Q*

Q	node removed from *Q*	Updated values of **s**(n)
1	1	$\mathbf{s}(2) = \begin{bmatrix} 5 \\ e \\ \varepsilon \end{bmatrix}$, $\mathbf{s}(3) = \begin{bmatrix} 3 \\ \varepsilon \\ e \end{bmatrix}$
2	2	$\mathbf{s}(3) = \begin{bmatrix} 9 \\ 4 \\ e \end{bmatrix}$
3	3	-

And the Kleene star of **A** can be computed as following

$$\mathbf{A}^* = \begin{bmatrix} \mathbf{s}(1)^T \\ \mathbf{s}(2)^T \\ \mathbf{s}(3)^T \end{bmatrix} = \begin{bmatrix} e & \varepsilon & \varepsilon \\ 5 & e & \varepsilon \\ 9 & 4 & e \end{bmatrix}.$$

5. INTERVAL ANALYSIS

This section generalizes the proposed algorithms to interval systems where inputs of the system specified with interval values.

An interval is defined by lower-bound and upper-bound values. Maximization and addition operators on intervals result a new interval with new lower and upper bounds which are analyzed in the first subsection. After that, a standard form of interval system is suggested. Then, proposed algorithms in previous chapters are expanded to interval systems, using dioid definition.

5.1.Operations on Intervals

This subsection analyzes maximization and addition operations on intervals.

5.1.1. Maximum of Intervals

If $x_1 \in I_1 = [a_1, b_1]$, $x_2 \in I_2 = [a_2, b_2]$, maximum of x_1 and x_2 belongs to a new interval, which its lower-bound and upper-bound are maximum lower-bounds and maximum upper-bounds of x_1 and x_2 respectively.

Two intervals are disjointed if there is no common area between them (Figure 10), enclosed if one is covered thoroughly by the other one (Figure 11) , and overlapped if they neither disjointed nor enclosed (Figure 12). The figures also show the interval that the $\max(x_1, x_2)$ lie in.

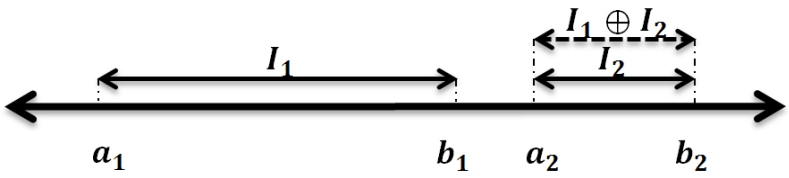

Figure 10: Disjointed intervals and their maximun

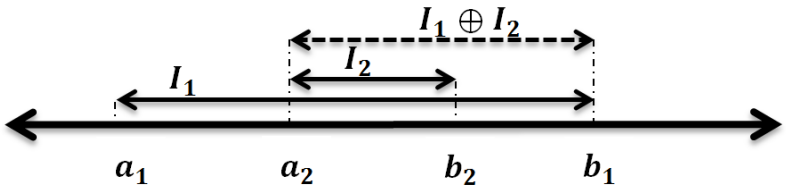

Figure 11: Enclosed intervals and their maximum

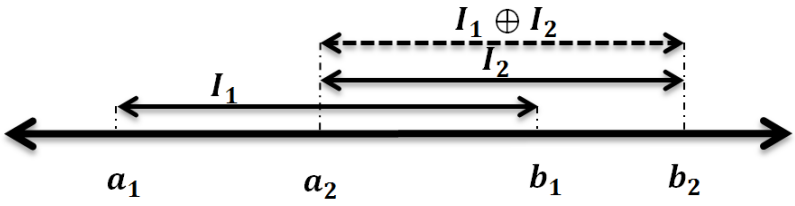

Figure 12: Overlapped intervals and their maximum

For the case that two intervals are disjointed, the right side interval is the maximum of intervals, because any point in there is higher than any point in left side interval. If two intervals are enclosed, the lower-bound and upper-bound of maximum interval is equal to the lower-bound of shorter interval and upper-bound of the other one, respectively. For overlapped intervals, the maximum is the same as disjointed intervals.

From mathematical perspective, since $a_1 \leq x_1 \leq b_1$ and $a_2 \leq x_2 \leq b_2$, following equations prove the intervals shown in the figures.

$$\begin{cases} x_1 \geq a_1 \\ and \\ x_2 \geq a_2 \end{cases} \rightarrow \begin{cases} \max(x_1, x_2) \geq a_1 \\ and \\ \max(x_1, x_2) \geq a_2 \end{cases} \rightarrow \max(x_1, x_2) \geq \max(a_1, a_2),$$

$$\begin{cases} x_1 \leq b_1 \\ and \\ x_2 \leq b_2 \end{cases} \rightarrow \begin{cases} \max(x_1, x_2) \leq b_1 \\ or \\ \max(x_1, x_2) \leq b_2 \end{cases} \rightarrow \max(x_1, x_2) \leq \max(b_1, b_2)\,,$$

hence

$$\max(a_1, a_2) \leq \max(x_1, x_2) \leq \max(b_1, b_2) \rightarrow x_1 \oplus x_2 \in [a_1 \oplus a_2, b_1 \oplus b_2]\,.$$

5.1.2. Addition of Intervals

If $x_1 \in I_1 = [a_1, b_1]$, $x_2 \in I_2 = [a_2, b_2]$, summation of x_1 and x_2 forms a new interval, which its lower-bound is equal to summation of lower-bounds of x_1 and x_2, and its upper-bound is equal to summation of upper-bounds of x_1 and x_2 (see figures 13 and 14).

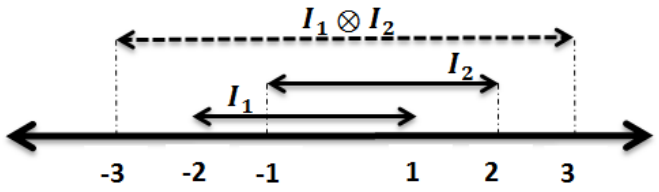

Figure 13: Summation of intervals

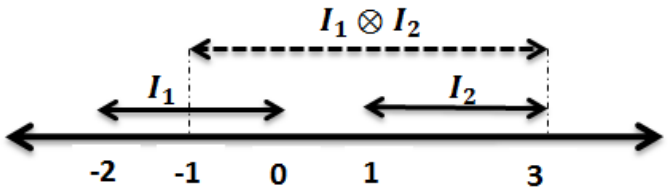

Figure 14:Summation of intervals

For the addition of intervals, there is no general graphical method to determine the lower-bound and upper-bound of the summation interval; however, they can be easily calculated by adding of lower-bounds and upper-bounds, respectively.

From mathematical point of view, since $a_1 \le x_1 \le b_1$ and $a_2 \le x_2 \le b_2$, following equations prove the aforementioned statement about lower-bound and upper-bound.

$$\begin{cases} a_1 \le x_1 \le b_1 \\ \quad and \\ a_2 \le x_2 \le b_2 \end{cases} \rightarrow \quad a_1 + a_2 \le x_1 + x_2 \le b_1 + b_2 \ ,$$

hence

$$x_1 \otimes x_2 \in [a_1 \otimes a_2, b_1 \otimes b_2] \ .$$

5.2.Interval System

This section shows that intervals form dioids. Since the proposed algorithms only require the properties of dioid, then they will apply to intervals directly.

Theorem 7.1:

Interval system on R_∞ with unit elements $e = [e, e]$ and ε=[ε, ε] , and operators $\oplus$ and $\otimes$ that are defined

$$I_1 \oplus I_2 = [a_1, b_1] \oplus [a_2, b_2] = [a_1 \oplus a_2 \ , b_1 \oplus b_2] \ ,$$

$$I_1 \otimes I_2 = [a_1, b_1] \otimes [a_2, b_2] = [a_1 \otimes a_2 \ , b_1 \otimes b_2] \ ,$$

is a dioid.

Proof:

1- Associativity of maximization

$$(I_1 \oplus I_2) \oplus I_3 = [a_1 \oplus a_2 , b_1 \oplus b_2] \oplus [a_3, b_3]$$

$$= [(a_1 \oplus a_2) \oplus a_3 , (b_1 \oplus b_2) \oplus b_3]$$

$$= [a_1 \oplus (a_2 \oplus a_3) , b_1 \oplus (b_2 \oplus b_3)]$$

$$= [a_1 , b_1] \oplus [a_2 \oplus a_3 , b_2 \oplus b_3]$$

$$= I_1 \oplus (I_2 \oplus I_3)$$

$$\forall I_1, I_2, I_3 \in R_{max}$$

∎

2- Commutativity of maximization

$$I_1 \oplus I_2 = [a_1 , b_1] \oplus [a_2 , b_2]$$

$$= [a_1 \oplus a_2 , b_1 \oplus b_2]$$

$$= [a_2 \oplus a_1 , b_2 \oplus b_1]$$

$$= [a_2 , b_2] \oplus [a_1 , b_1]$$

$$= I_2 \oplus I_1 .$$

$$\forall I_1, I_2 \in R_{max}$$

∎

3- Associativity of addition

$$(I_1 \otimes I_2) \otimes I_3 = [a_1 \otimes a_2 , b_1 \otimes b_2] \otimes [a_3 , b_3]$$

$$= [(a_1 \otimes a_2) \otimes a_3 , (b_1 \otimes b_2) \otimes b_3]$$

$$= [a_1 \otimes (a_2 \otimes a_3), b_1 \otimes (b_2 \otimes b_3)]$$

$$= I_1 \otimes (I_2 \otimes I_3).$$

$$\forall\, I_1, I_2, I_3 \in R_{max}$$

■

4- Distributivity of addition with respect to maximization

$$(I_1 \oplus I_2) \otimes I_3 = [a_1 \oplus a_2 \,, b_1 \oplus b_2] \otimes [a_3 \,, b_3]$$

$$= [(a_1 \oplus a_2) \otimes a_3 \,, (b_1 \oplus b_2) \otimes b_3]$$

$$= [(a_1 \otimes a_3) \oplus (a_2 \otimes a_3) \,, (b_1 \otimes b_3) \oplus (b_2 \otimes b_3)]$$

$$= [a_1 \otimes a_3 \,, b_1 \otimes b_3] \oplus [a_2 \otimes a_3 \,, b_2 \otimes b_3]$$

$$= (I_1 \otimes I_3) \oplus (I_2 \otimes I_3).$$

$$\forall\, I_1, I_2, I_3 \in R_{max}$$

Since addition is not assumed to be commutative, second part of this section needs to be proved.

$$I_3 \otimes (I_1 \oplus I_2) = [a_3 \,, b_3] \otimes [a_1 \oplus a_2 \,, b_1 \oplus b_2]$$

$$= [a_3 \otimes (a_1 \oplus a_2) \,, b_3 \otimes (b_1 \oplus b_2)]$$

$$= [(a_3 \otimes a_1) \oplus (a_3 \otimes a_2) \,, (b_3 \otimes b_1) \oplus (b_3 \otimes b_2)]$$

$$= [a_3 \otimes a_1 \,, b_3 \otimes b_1] \oplus [a_3 \otimes a_2 \,, b_3 \otimes b_2]$$

$$= (I_3 \otimes I_1) \oplus (I_3 \otimes I_2).$$

$$\forall\, I_1, I_2, I_3 \in R_{max}$$

■

5- Existence of a null element

$$I \oplus \varepsilon = [a\,, b] \oplus [\varepsilon, \varepsilon]$$

$$= [a \oplus \varepsilon\,, b \oplus \varepsilon]$$

$$= [a\,, b]$$

$$= I.$$

$$\exists \varepsilon \in R_{max}: \forall\, I\ \in R_{max}$$

■

6- Absorbing null element

$$I \otimes \varepsilon = [a\,, b] \otimes [\varepsilon\,, \varepsilon]$$

$$= [a \otimes \varepsilon\,, b \otimes \varepsilon]$$

$$= [\varepsilon\,, \varepsilon]$$

$$= \varepsilon.$$

$$\forall\, I\ \in R_{max}$$

Since addition is not assumed to be commutative, second part of this section needs to be proved.

$$\varepsilon \otimes I = [\varepsilon\,, \varepsilon] \otimes [a\,, b]$$

$$= [\varepsilon \otimes a\,, \varepsilon \otimes b]$$

$$= [\varepsilon\,, \varepsilon]$$

$$= \varepsilon.$$

$$\forall\, I\ \in R_{max}$$

■

7- Existence of an identity element

$$I \otimes e = [a\,, b] \otimes [e\,, e]$$

$$= [a \otimes e\,, b \otimes e]$$

$$= [a\,, b]$$

$$= I.$$

$$\exists e \in R_{max}: \forall\, I\ \in R_{max}$$

Since addition is not assumed to be commutative, second part of this section needs to be proved.

$$e \otimes I = [e\,, e] \otimes [a\,, b]$$

$$= [e \otimes a\,, e \otimes b]$$

$$= [a\,, b]$$

$$= I.$$

$$\exists e \in R_{max}: \forall\, I\ \in R_{max}$$

■

8- Idempotency of addition

$$I \oplus I = [a\,, b] \oplus [a\,, b]$$

$$= [a \oplus a\,, b \oplus b]$$

$$= [a\,, b]$$

$$= I.$$

$$\forall\, I\ \in R_{max}$$

■

As it is mentioned in dioid subsection of background chapter, max-plus algebra is a dioid. Therefore, proposed longest path algorithms can be expanded to any other dioids. This chapter proved that defined interval system is a dioid. Therefore, proposed system matrix and Kleene star algorithms are applicable for interval systems as well.

5.3.Numerical Example

Figure 15 represents the same system shown in Figure 1, however, the edge weights are changed to interval values. Terminating node of each job is connected to a dummy node (shown by dashed lines).

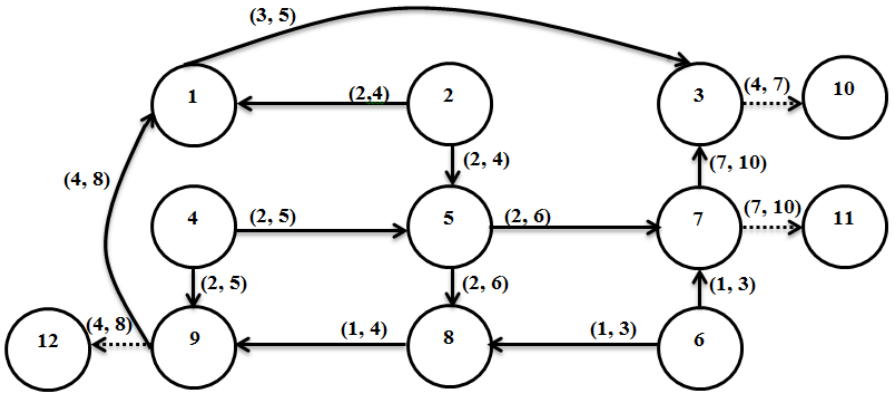

Figure 15: Manufacturing system with interval weights

Initialization step of algorithm 2 for the given example is similar to the numerical example in system matrix chapter. Table 8 shows the initialization phase.

Table 8: Initialization phase of algorithm 2

Node	1	2	3	4	5	6	7	8	9
Initial Value for **s**(n)	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(e,e)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(e,e)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(e,e)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$	$\begin{bmatrix}(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$

The results are shown in the Table 9.

Table 9: Summary of the results

Q	node removed from Q	Updated values of **s**(n)
2,4,6	2	**s**(1) = $\begin{bmatrix}(2,4)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$, **s**(5)= $\begin{bmatrix}(2,4)\\(\varepsilon,\varepsilon)\\(\varepsilon,\varepsilon)\end{bmatrix}$
4,6	4	**s**(5)= $\begin{bmatrix}(2,4)\\(2,5)\\(\varepsilon,\varepsilon)\end{bmatrix}$, **s**(9)= $\begin{bmatrix}(\varepsilon,\varepsilon)\\(2,5)\\(\varepsilon,\varepsilon)\end{bmatrix}$
5,6	5	**s**(7)= $\begin{bmatrix}(4,10)\\(4,11)\\(\varepsilon,\varepsilon)\end{bmatrix}$, **s**(8)= $\begin{bmatrix}(4,10)\\(4,11)\\(\varepsilon,\varepsilon)\end{bmatrix}$
6	6	**s**(7) = $\begin{bmatrix}(4,10)\\(4,11)\\(1,3)\end{bmatrix}$, **s**(8)= $\begin{bmatrix}(4,10)\\(4,11)\\(1,3)\end{bmatrix}$
7,8	7	**s**(3)= $\begin{bmatrix}(11,20)\\(11,21)\\(8,13)\end{bmatrix}$, **s**(11)= $\begin{bmatrix}(11,20)\\(11,21)\\(8,13)\end{bmatrix}$
8	8	**s**(9)= $\begin{bmatrix}(5,14)\\(5,15)\\(2,7)\end{bmatrix}$
9	9	**s**(1)= $\begin{bmatrix}(9,22)\\(9,23)\\(6,15)\end{bmatrix}$, **s**(12)= $\begin{bmatrix}(9,22)\\(9,23)\\(6,15)\end{bmatrix}$

1	1	$\mathbf{s}(3)=\begin{bmatrix}(12,27)\\(12,28)\\(9,20)\end{bmatrix}$
3	3	$\mathbf{s}(10)=\begin{bmatrix}(16,34)\\(16,35)\\(13,27)\end{bmatrix}$

And finally, the system matrix is

$$\mathbf{A}=\begin{bmatrix}\mathbf{s}(10)^{\mathbf{T}}\\\mathbf{s}(11)^{\mathbf{T}}\\\mathbf{s}(12)^{\mathbf{T}}\end{bmatrix}=\begin{bmatrix}(16,34) & (16,35) & (13,27)\\(11,20) & (11,21) & (8,13)\\(9,22) & (9,23) & (6,15)\end{bmatrix}.$$

6. CONCLUSION

This thesis proposed an efficient algorithm that calculates start times in a manufacturing system with computational complexity of $O(16|N|)$ where $|N|$ is number of nodes in the graph of the system.

Next, the former algorithm is modified to calculate the system matrix of manufacturing systems. System matrix can be used for performance measures such as completion time, lateness, tardiness, makespan, and period of the system. The thesis proposed an efficient and structured algorithm to calculate the system matrix with computational complexity of $O(5J|N| + 11|\mathrm{N}|)$ where J is number of jobs and $|N|$ is number of nodes in the DAG.

The thesis also proposed an efficient method to compute the Kleene star operator for systems with DAG structures. Considering other algorithms, the proposed algorithm runs slightly faster than the fastest algorithm in literature (see Table 5).

Finally, the thesis defined an interval system based on the concept of dioids, and then expanded the proposed algorithms in previous steps to interval systems. In interval systems, input data are shown by interval values.

7. FUTURE STUDY

Following subjects can be considered as extension to this thesis:

- Perturbation analysis in different cases such as processing times, and topology of the graph.
- Considering matrix form of manufacturing systems and modifying proposed algorithms accordingly.
- Focusing on specific types of manufacturing systems with specific graph structures and modifying the algorithms.
- Assigning certain distribution functions for input data and expanding the research scope to stochastic environment.

REFERENCES

[1] A. B. Kahn, "Topological sorting of large networks," *Communications of the ACM,* vol. 5, pp. 558-562, Nov. 1962.

[2] Z. Li, N. Wu, and M. Zhou, "Deadlock control of automated manufacturing systems based on Petri nets—A literature review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 42, pp. 437-462, July 2012.

[3] A. Imaev and R. P. Judd, "Block diagram-based modeling of manufacturing systems using," in *American Control Conference, 2009. ACC'09.*, 2009, pp. 4711-4716.

[4] M. Zhou, F. DiCesare, and A. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *Robotics and Automation, IEEE Transactions on,* vol. 8, pp. 350-361, Jun 1992.

[5] A. Imaev and R. P. Judd, "Hierarchial modeling of manufacturing systems using max-plus algebra," in *American Control Conference, 2008*, 2008, pp. 471-476.

[6] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and linearity: an algebra for discrete event systems*: John Wiley & Sons Ltd, 1992.

[7] G. Mejía and C. Montoya, "Scheduling manufacturing systems with blocking: a Petri net approach," *International Journal of Production Research,* vol. 47, pp. 6261-6277, 2009.

[8] K. Labadi, "Contribution to modelling and performances analysis of logistic systems by using a new stochastic Petri nets model," 2005.

[9] B. Heidergott, G. J. Olsder, and J. Van Der Woude, *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*: Princeton University Press, 2014.

[10] M. Akian, R. Bapat, and S. Gaubert, "Max-plus algebra," *Handbook of linear algebra. Chapman and Hall, London,* 2006.

[11] J.-L. Bouquard, C. Lenté, and J.-C. Billaut, "Application of an optimization problem in max-plus algebra to scheduling problems," *Discrete Applied Mathematics,* vol. 154, pp. 2064-2079, 2006.

[12] H. Takahashi, H. Goto, and M. Kasahara, "Application of a critical chain project management based framework on max-plus linear systems," in *Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09. International Conference on*, 2009, pp. 898-903.

[13] M. Singh and R. P. Judd, "Efficient calculation of the makespan for job-shop systems without recirculation using max-plus algebra," *International Journal of Production Research,* vol. 52, pp. 5880-5894, 2014.

[14] M. Singh, "Mathematical Models, Heuristics and Algorithms for Efficient Analysis and Performance Evaluation of Job Shop Scheduling Systems Using Max-Plus Algebraic Techniques," Ph.D. dissertation, Indust. & Sys. Eng., Ohio Univ., Athens, OH, 2013.

[15] P. R. Patlola, "Efficient Evaluation of Makespan for a Manufacturing System Using Max-Plus Algebra," M.S. thesis, Indust. & Sys. Eng., Ohio Univ., Athens, OH, 2011.

[16] A. Imaev and R. P. Judd, "Spectral properties for the max plus dynamics matrix for flow shops," in *Proceedings of the Ninth IASTED International Conference on Control and Applications*, 2007, pp. 110-115.

[17] A. Nambiar and R. P. Judd, "Max-plus-based mathematical formulation for cyclic permutation flow-shops," *International Journal of Mathematical Modelling and Numerical Optimisation,* vol. 2, pp. 85-97, 2011.

[18] L. Houssin, "Cyclic jobshop problem and (max, plus) algebra," in *World IFAC Congress (IFAC 2011)*, 2011, pp. 2717-2721.

[19] A. Seleim and H. ElMaraghy, "Max-plus Modeling of Manufacturing Flow Lines," *Procedia CIRP,* vol. 17, pp. 71-75, 2014.

[20] A. Seleim and H. ElMaraghy, "Generating max-plus equations for efficient analysis of manufacturing flow lines," *Journal of Manufacturing Systems,* 2014. Available http://dx.doi.org/10.1016/j.jmsy.2014.07.002

[21] H. Kajiwara, Y. Hitoi, and K. Hassan, "Max-plus approach to scheduling problems of block assembly lines," in *Proc Int Conf Mar Technol*, 2010, pp. 189-194.

[22] H. Kajiwara, Y. Hitoi, and Y. Nakao, "Max-plus-algebra based scheduling of a ship building line," in *Proc Int Conf Comput Appl Shipbuild*, 2009, pp. 149-155.

[23] K. Hiroyuki, Y. Hitoi, and Y. Nakao, "On scheduling a shipbuilding line based on Max-Plus system dynamic representation," in *ICCAS-SICE, 2009*, 2009, pp. 1738-1741.

[24] T. J. van den Boom, G. D. Lopes, and B. De Schutter, "A modeling framework for model predictive scheduling using switching max-plus linear models," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 2013, pp. 5456-5461.

[25] I. Nasri, G. Habchi, and R. Boukezzoula, "Use of (max,+) algebra for scheduling and optimization of HVLV systems subject to preventive maintenance," *Simulation Modelling Practice and Theory,* vol. 46, pp. 149-163, 2014.

[26] M. Alirezaei, T. van den Boom, and R. Babuska, "Max-plus algebra for optimal scheduling of multiple sheets in a printer," in *American Control Conference (ACC), 2012*, 2012, pp. 1973-1978.

[27] W. H. Fleming, "Max-plus stochastic processes," *Applied Mathematics and Optimization,* vol. 49, pp. 159-181, 2004.

[28] B. Heidergott, "A characterisation of (max,+)-linear queueing systems," *Queueing systems,* vol. 35, pp. 237-262, 2000.

[29] N. K. Krivulin, "Max-plus algebra models of queueing networks," *arXiv preprint arXiv:1212.0578,* 2012.

[30] N. K. Krivulin, "The max-plus algebra approach in modelling of queueing networks," *arXiv preprint arXiv:1212.0895,* 2012.

[31] N. K. Krivulin, "Algebraic modelling and performance evaluation of acyclic fork-join queueing networks," in *Advances in Stochastic Simulation Methods*, ed: Springer, 2000, pp. 63-81.

[32] S. Masuda, "Monitoring and scheduling methods for MIMO-FIFO systems utilizing max-plus linear representation," *IEMS,* vol. 7, pp. 23-33, 2008.

[33] S. Masuda, "Derivation algorithm of state-space equation for production systems based on max-plus algebra," *IEMS,* vol. 3, pp. 1-11, 2004.

[34] T. Petrović and S. Bogdan, "Matrix-based sequencing in multiple re-entrant flowlines," *Transactions of the Institute of Measurement and Control,* vol. 33, pp. 359-385, 2011.

[35] H. TAKAHASHI, "Efficient representation of the state equation in max-plus linear systems with interval constrained parameters," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,* vol. 95, pp. 608-612, 2012.

[36] G. Cohen, P. Moller, J.-P. Quadrat, and M. Viot, "Algebraic tools for the performance evaluation of discrete event systems," *Proceedings of the IEEE,* vol. 77, pp. 39-85, 1989.

[37] H. Goto and M. Kasahara, "Efficient Computation Methods for the Kleene Star in Max-Plus Linear Systems," in *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, 2009, pp. 1388-1393.

[38] H. TAKAHASHI, "Fast computation methods for the Kleene star in max-plus linear systems with a DAG structure," *IEICE transactions on fundamentals of electronics, communications and computer sciences,* vol. 92, pp. 2794-2799, 2009.

[39] H. Goto, "High-Speed Computation of the Kleene Star in Max-Plus Algebraic System Using a Cell Broadband Engine," *IEICE transactions on information and systems,* vol. 93, pp. 1798-1806, 2010.

[40] H. Goto, "Acceleration of computing the Kleene Star in Max-Plus algebra using CUDA GPUs," *IEICE transactions on information and systems,* vol. 94, pp. 371-374, 2011.

[41] S. Skiena, "Dijkstra's Algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley,* pp. 225-227, 1990.

[42] D. B. Johnson, "A note on Dijkstra's shortest path algorithm," *Journal of the ACM (JACM),* vol. 20, pp. 385-388, 1973.

[43] R. Bellman, "On a routing problem," DTIC Document1956.

[44] L. R. Ford Jr, "Network flow theory," DTIC Document1956.

[45] E. F. Moore, *The shortest path through a maze*: Bell Telephone System., 1959.

[46] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM,* vol. 5, p. 345, 1962.

[47] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Operations research,* vol. 17, pp. 395-412, 1969.

[48] M. Er, "A parallel computation approach to topological sorting," *The Computer Journal,* vol. 26, pp. 293-295, 1983.

[49] A. D. Kalvin and Y. L. Varol, "On the generation of all topological sortings," *Journal of Algorithms,* vol. 4, pp. 150-162, 1983.

[50] D. J. Pearce and P. H. Kelly, "A dynamic algorithm for topologically sorting directed acyclic graphs," in *Experimental and Efficient Algorithms*, ed: Springer, 2004, pp. 383-398.

[51] Y. Inoue and S.-i. Minato, "An Efficient Method for Indexing All Topological Orders of a Directed Graph," in *Algorithms and Computation*, ed: Springer, 2014, pp. 103-114.

[52] G. F. Italiano, "Finding paths and deleting edges in directed acyclic graphs," *Information Processing Letters,* vol. 28, pp. 5-11, 1988.

[53] D. Jungnickel and T. Schade, *Graphs, networks and algorithms*: Springer, 2008.

[54] R. E. Moore, *Interval analysis* vol. 4: Prentice-Hall Englewood Cliffs, 1966.

[55] H. Schichl and A. Neumaier, "Interval analysis on directed acyclic graphs for global optimization," *Journal of Global Optimization,* vol. 33, pp. 541-562, 2005.

[56] R. E. Moore and R. Moore, *Methods and applications of interval analysis* vol. 2: SIAM, 1979.

Thesis and Dissertation Services